Symmetries, Local Names and Dynamic
(De)-allocation of Names

Vincenzo Ciancia\textsuperscript{1,2} Ugo Montanari\textsuperscript{1}

Abstract

The semantics of name-passing calculi is often defined employing coalgebraic models over presheaf categories. This elegant theory lacks finiteness properties, hence it is not apt to implementation. Coalgebras over named sets, called history-dependent automata, are better suited for the purpose due to locality of names. A theory of behavioural functors for named sets is still lacking: the semantics of each language has been given in an ad-hoc way, and algorithms were implemented only for the $\pi$-calculus. Existence of the final coalgebra for the $\pi$-calculus was never proved. We introduce a language of accessible functors to specify history-dependent automata in a modular way, leading to a clean formulation and a generalisation of previous results, and to the proof of existence of a final coalgebra in a wide range of cases.

1 Introduction

At the end of the eighties, it was recognised that existing models of communication and concurrency, such as Petri nets \cite{36} or the CCS \cite{29}, were not suitable to model dynamic reconfiguration of the structure being considered. To give to the CCS the ability to change the communication network at runtime, the $\pi$-calculus \cite{30} was invented. The key feature of the calculus is the combination of fresh name generation and name passing. These turned out to play a crucial role in many other contexts (e.g. secure communication and nonces \cite{1}, causality \cite{7}, sessions in service-oriented computing) that pervade modern computer science.

Ordinary set-theory does not model name generation in a fully satisfactory way: in particular, definitions such as bisimulation become non-standard due to the presence of allocation of new names. To overcome these limitations, in

\textsuperscript{1} Research partially supported by the EU FP6-IST IP 16004 project SENSORIA
\textsuperscript{2} Research partially supported by the Comunidad de Madrid program PROMESAS (S-0505/TIC/0407).
the nineties there was a focus shift from ordinary labelled transition systems to coalgebras over presheaf categories, or over categories of algebras, to model the syntax \cite{18,21} and the semantics \cite{10,32,20} of name passing.

Elegant formalisations and nice full abstractness results are the strong point of all these models, that however lack the ability to discard no longer used fresh names. Algorithms that manipulate the semantics (e.g. minimisation and model checking) typically work by reducing the system to finite states. In absence of garbage collection, finite state systems are obtained only for very simple processes. The problem of doing static analysis in the presence of fresh name generation constructs is common in the process calculi and service oriented computing communities (for example, in calculi that can handle sessions and session identifiers). Typically, an approximation of the semantics using only a finite set of names is used. However, the necessity to fix the number of resources before the analysis limits the applicability of this methodology. Similar issues arise when doing static analysis of programming languages that feature dynamic memory allocation (e.g. C/C++ or Java).

The category of named sets and its coalgebras, history-dependent automata (HD-automata, \cite{34,35}), were invented as a formal methodology to reason about such systems, and to implement finite-state algorithms dealing with resource allocation. The most important property of named sets is that the morphisms implicitly perform garbage collection of unused names. Earlier versions of HD-automata simply featured finite sets of names attached to each element. This was sufficient to represent the semantics of Petri nets and the CCS, but attempting to work on the $\pi$-calculus resulted in a model that was not fully abstract. The introduction of symmetries in named sets was the key to define a minimisation procedure for the $\pi$-calculus using HD-automata \cite{16,17}, that, remarkably, is able to compute the best symmetry reduction of a system up-to bisimulation. The importance of modelling symmetries for concurrency is becoming an established fact also because of algorithmic benefits due to efficiency of the representation (see e.g. \cite{38} and related works). Symmetries are a source of increased complexity in the definition of named sets, and it is necessary to develop a simple, modular specification language for functors in this category, in order to allow the theory to be reused in various contexts.

The garbage collection properties of named sets mostly come from the fact that names are considered as local, rebindable resources, rather than fixed, global constants as in the $\pi$-calculus. This is done by quotienting over an equivalence relation identifying all elements that can be obtained from each other by an injective renaming. A number of new developments, that we present in this work, come from the study of how typical operations that are used to model the semantics of programming languages can be specified with local names, that is, working on the canonical representatives of such an equivalence. These results are not only useful for their algorithmic properties, but also because
they describe a natural model of systems that do not have a global “naming authority”, (e.g. peer-to-peer systems, or self-organising sensor networks). As an example, the categorical product of named sets introduces at a formal level the machinery to establish a binding of names between two systems (having local names) that are put in a relationship.

We can identify at least four important constructs in name passing:

1. Dynamic allocation of names.
2. Deallocation of fresh names that will no longer be used.
3. Composition of systems into a larger one.
4. Non-deterministic choice quantified over an arbitrary name (be it fresh or already known).

Point 1 was solved from a theoretical point of view using presheaf categories indexed over names. Point 2 allows one to implement algorithms that manipulate the semantics (for example, minimisation up to bisimulation, or static analysis techniques such as model checking or trace analysis). Point 3 arises when trying to put together different systems that use names. If names are global, like in the π-calculus, this step is rather trivial: one can simply assume that equal names in the two systems have the same meaning. A practical example is the world wide web. There, we find a naming authority that ensures that names (in this context, URLs) have the same meaning in all possible sites. On the other hand, if this assumption can not be made, one has to establish a binding between the local names of two possible systems. Point 4 comes out when a non-deterministic choice of one name is required (e.g. in modelling input operations, or method calls of object-oriented languages), and it is not known whether the chosen name is fresh or already known. In this case, we have an infinitely branching operational semantics which is not directly implementable. One might be tempted to say that the solution of this problem is simple: just represent all the infinite fresh names using a placeholder. However, as we explain in detail in §6.5, such a representation is not correct if we cannot tell whether a name is semantically fresh or not, that is, if it is actually used by the system. This is the case in the π-calculus, where there can be redundant names, that will never be observed, and it may as well happen in any Turing-equivalent programming language featuring global variables.

In category theory, operations are specified by functors. In presheaf categories the functors implementing the aforementioned key constructs are well known; in [20] a language of accessible endofunctors over Set^I and Set^F can be found. In this work, we show how these functors are defined in named sets. This sheds light on various common practices that are typically used when

---

3 Here F stands for the category of finite cardinals and all functions, and I stands for the subcategory only having injective functions. For simplicity, we will consider I as the equivalent category of finite subset of natural numbers and injective functions.
working with local names, and were implicitly adopted in previous works. A first step was done in [23] and [19], where it is proved that the categories of nominal sets, permutation algebras, named sets and the Schanuel topos \( \text{Sh}(\text{Top}) \) (the full subcategory of pullback preserving presheaves in \( \text{Set}^I \)) are all equivalent. In this work, we extend these equivalence results from the base categories to categories of coalgebras, by defining over named sets a compositional language of endofunctors admitting a final coalgebra. All the functors that we define are accessible, therefore they have a final coalgebra. Moreover, since accessibility is a compositional property, this result propagates to all the functors definable in the framework.

After introducing the required background in \( \S \) 2, we start by defining the functors in the category of permutation algebras in \( \S \) 3. Product and coproduct are lifted from the definitions in \( \text{Set} \). For the countable power set, particular care is required, since this functor does not restrict to the finitely supported case. The definition that we give of the name abstraction functor is the first result we present. We show with an example that initial algebras for the obtained functor exactly give De Bruijn indexes. If compared to the name abstraction of [22], this shows that De Bruijn indexes and \( \alpha \)-conversion are linked by a categorical isomorphism, hence they are at the same conceptual level.

In the second part of the paper, we focus on named sets, presented in \( \S \) 4. A “notational” contribution of this work, in the light of simplifying and generalising the presentation of HD-automata, is a structured definition of named sets. Similarly to [19], we introduce a category modelling name mappings in the presence of symmetry, that we use to define the product and power set.

We define the functors for name passing exploiting the results in \( \S \) 3 and the equivalence result of [23]. The name abstraction functor (\( \S \) 5, presented in [11]), provides a way to allocate new names, and implicitly deallocates them, addressing the problem of garbage collection and providing the operations required by points 1 and 2 above.

In \( \S \) 6.1 we show a canonical way to represent partial mappings between symmetries. The categorical product of named sets (\( \S \) 6.2) is not made up of pairs, like in permutation algebras or presheaves. Instead, for each two elements, there is a name mapping between them. This is a common situation in computing (point 3 above): whenever two extraneous entities are related to each other (e.g. two isolated networks that have to be interconnected), it is necessary to bind their common resources (e.g. addresses of nodes) establishing a correspondence on a case by case basis. It is surprising that this very general observation just comes from the standard construction of the product. In \( \S \) 6.3 we also present the coproduct, having a simple definition, for completeness.

Point 4 is addressed by the definition of the countable power set (\( \S \) 6.4). In
[17], a normalisation step was employed in the action of the functor on arrows to solve the problem of redundant names that is introduced by the infinitary input action of the early $\pi$-calculus. We show in §6.5 that this corresponds to a finite representation of a subfunctor of the power set, thus proving the correctness of that approach.

We prove that the definitions over named sets are equivalent (as functors, in the categorical sense) to the corresponding notions in permutation algebras. This enables us in §7 to propagate accessibility results, and to lift the equivalence on the base categories to the corresponding categories of coalgebras, thus providing a final coalgebra theorem.

As an example, we show in §8.1 that the various techniques introduced in [17], which are ad-hoc for the $\pi$-calculus, are included in our framework, thus they can be reused for other nominal calculi. As a consequence of the definition of the product, bisimulation becomes a ternary relation, employing two states, and a name mapping between them. This is a common denominator among all the different versions of HD-automata, but it was not realised before that it is just the ordinary notion of bisimulation, with name mappings arising from the product. A consequence is recovering history-dependent (causal) bisimulation for place-transition Petri nets as a standard definition (§8.2).

2 Background

Here we introduce the grounds of our work, that is, nominal sets as a category of algebras, theory morphisms, multi-coproducts and coalgebras of accessible endofunctors. Hereafter, $|C|$ denotes the objects of a category or a diagram $C$, $\hom(c_1, c_2)$ the homset of $c_1$ and $c_2$, and $\langle t_i \rangle_{i \in I}$ a tuple indexed by a set $I$.

2.1 Nominal Sets as Algebras for the Permutation Signature

We introduce the theory of sets equipped with a notion of action of a permutation over the set of natural numbers $\omega$, and of their full subcategory of nominal sets, employed by Gabbay and Pitts to model name binding in [21] and subsequent works. FM-sets can be defined as algebras for the finite-kernel permutation group viewed as an algebraic theory. These permutation algebras were employed to model the semantics of name passing calculi [32,8,33], even involving name fusions [9]. Using the algebraic definition, equivariant functions are the algebra homomorphism, and powerful tools from the well established theory of algebras can be reused (e.g. theory morphisms in §3.1). The most important definitions are those of support, symmetry and orbits.
The set $\text{Autf}$ of finite-kernel permutations over the set of natural numbers $\omega$, that is, those permutations only affecting a finite number of elements, forms a group whose operation is composition. The permutation signature is the set $\text{Autf}$ considered as the one-sorted equational theory\footnote{We call it a “signature” to be consistent with previous works.} $\{\pi : 1 \to 1 \mid \pi \in \text{Autf}\}$ with axioms $id(x) = x$ and $\pi_1(\pi_2(x)) = (\pi_1 \circ \pi_2)(x)$.

**Definition 2.1 (permutation algebra)** A permutation algebra is an algebra $A = \langle A, \{\pi_A : A \to A \mid \pi \in \text{Autf}\} \rangle$ for $\text{Autf}$, where $A$ is the carrier set, and $\pi_A$ is the interpretation of $\pi$, also called permutation action. An equivariant function is an algebra homomorphism from $A$ to $B$, that is, a function $f : A \to B$ such that $\forall \pi \in \text{Autf} . f(\pi_A(x)) = \pi_B(f(x))$.

Hereafter, we denote the underlying set of a permutation algebra $A$ (resp. $B$) with $A$ (resp. $B$). The set of natural numbers $\omega$ can be considered a permutation algebra using the natural interpretation $\pi_\omega = \pi$.

**Definition 2.2 (symmetry)** Given a permutation algebra $A$, the symmetry of an element $a \in A$ is the set of all permutations fixing $a$ in $A$, defined as $G_A(a) = \{ \pi \in \text{Autf} \mid \pi_A(a) = a \}$.

**Definition 2.3 (support)** Let $\text{fix}(X)$ denote the set $\{ \pi \in \text{Autf} \mid \pi|_X = id_X \}$. We say that $X \subseteq \omega$ supports $a \in A$ if $\text{fix}(X) \subseteq G_A(a)$, that is, all permutations fixing $X$ also fix $a$ in $A$. The least such finite set $X$, if it exists, is called the support of $a$, written $\text{supp}_A(a)$. We call a permutation algebra finitely supported, or a nominal set, if it only contains finitely supported elements.

Each element of a permutation algebra is trivially supported by $\omega$. A finite supporting set might not exist. If there is one, the support is the intersection of all of them. The notion of support generalises that of “free names”, thus we will often refer to $\omega$ as the set of names. The notion of symmetry models indistinguishability of free names with respect to certain permutations.

**Definition 2.4 (category of permutation algebras)** Permutation algebras and their morphisms form a category, named $\text{Alg}^\pi$. We denote with $\text{FSAlg}^\pi$ the full subcategory of finitely supported permutation algebras.

Equivalently, $\text{Alg}^\pi$ is the functor category $\text{Set}^{\text{Autf}}$ where $\text{Autf}$ is the one-object groupoidal category whose arrows are finite kernel permutations.

**Definition 2.5 (orbit)** The orbit of $a \in A$ is $\text{orb}_A(a) = \{ \pi_A(a) \mid \pi \in \text{Autf}\}$.

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

Example 2.6 (terms with variables) Let $\Sigma$ be a signature. Terms with variables in $\omega$ form a permutation algebra $T = \langle T_\Sigma(\omega), \{\pi_T\} \rangle$ having permutation action $\pi_T(t) = t[\pi(i)/i]$ for $i \in \omega$. It is easy to see that a finite set $X \subseteq \omega$ supports a term $t$ iff its set of free variables $FV(t)$ is a subset of $X$. The least such set, or the support, is the set of free variables of $t$.

Next, we survey a number of known or folklore results that are needed.

Remark 2.7 In a finitely supported permutation algebra $A$, for each $a \in A$, we have $\pi|_{\text{supp}(a)} = \pi'_|{\text{supp}(a)} \implies \pi_A(a) = \pi'_A(a)$. Because of this, we usually define a permutation $\pi$ only on the support of an element $a$, when it is clear 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. A concrete definition may be given by fixing a choice from the infinite set of permutations that agree with $\pi$ on $\text{supp}(a)$ (e.g. exploiting an order on names).

The following theorem asserts that the symmetry may grow, and does not shrink, along morphisms. From this, the support never grows along morphisms. This leads, in all categorical formalisms that handle names using injective relabellings (presheaves, nominal sets and named sets), to the necessity of defining specialised functors for name abstraction.

Theorem 2.8 For each $f : \langle A, \{\pi_A\} \rangle \to \langle B, \{\pi_B\} \rangle$, and $a \in A$, it holds that $G_A(a) \subseteq G_B(f(a))$. Therefore, $\text{supp}_B(f(a)) \subseteq \text{supp}_A(a)$.

The following “isomorphism theorem” is of fundamental importance for named sets, since it asserts that a named set represents a class of isomorphic permutation algebras, as we will see in §4.

Theorem 2.9 Two permutation algebras $A$ and $B$ are isomorphic iff there is a choice of canonical representatives $A^o$ and $B^o$ for them, and an isomorphism $i : A^o \to B^o$ in Set, such that for all $a^o \in A^o$ it holds $G_A(a^o) = G_B(i(a^o))$.

Finally, the following theorem gives a finite representation of the symmetry of finitely supported permutation algebras. The infinite set of all permutations in $G_A(a)$ is obtained from the finite set of permutations in $G_A(a)$ that only alter $\text{supp}(a)$, by composition with all the permutations that only alter names outside $\text{supp}(a)$. This is used in named sets to get a finite description of $G(a)$.

Theorem 2.10 The symmetry $G_A(a)$ of $a \in A$ is obtained by composition of two subgroups: $G_A(a) = fix(\text{supp}_A(a)) \circ (G_A(a) \cap fix(\omega \setminus \text{supp}_A(a)))$. 

7
2.2 Theory Morphisms

Theory morphisms, or views, are equation-preserving signature morphisms $M : \Sigma_1 \rightarrow \Sigma_2$, that yield algebras of $\Sigma_1$ from algebras of $\Sigma_2$ (see [25]).

A signature morphism $M$ between $\Sigma_1$ and $\Sigma_2$ is a function from the operators of $\Sigma_1$ to those of $\Sigma_2$ that respects operator arity. $M$ is inductively extended to $T_{\Sigma}(V)$ (the free $\Sigma$-algebra over a set of variables $V$) 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 the equations derivable from $E_1$. 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. A definition of the left adjoint is not required here, thus it is omitted.

**Definition 2.11 (forgetful functor)** Let $Th_1 = \langle \Sigma_1, E_1 \rangle$ and $Th_2 = \langle \Sigma_2, E_2 \rangle$ be two specifications. A theory morphism $M : Th_1 \rightarrow Th_2$ associates to a $Th_2$-algebra $A = \langle A, \{op_A \mid op \in \Sigma_2\} \rangle$ a $Th_1$-algebra $U(A) = \langle A, \{op_{U(A)} \mid op \in \Sigma_1\} \rangle$ with the same carrier, where $op_{U(A)} = M(op)_A$ for each operator $op$ in $\Sigma_1$. The map $U$ extends to a functor, acting on arrows as $U(f) = f$.

2.3 Coalgebras, Bisimulation and Accessibility

Coalgebras (for an introduction, see [26,37,2]) are a mathematical model based on the idea that equivalence should not be determined by syntactic equality, but rather by the observable properties of a system.

**Definition 2.12 (coalgebra)** For $T$ an endofunctor over $C$, a $T$-coalgebra is an arrow $f : a \rightarrow T(a)$. Given two coalgebras $f : a \rightarrow T(a)$ and $g : b \rightarrow T(b)$, a coalgebra morphism is an arrow $h : a \rightarrow b$ such that $T(h) \circ f = g \circ h$.

$T$-coalgebras and their morphisms form the category $\text{Coalg}(T)$. For example, labelled transition systems are $\mathcal{P}_{\text{fin}}(\mathcal{L} \times -)$-coalgebras in $\text{Set}$. The most important coalgebraic notion is bisimulation. The definition requires products.

**Definition 2.13 (bisimulation)** Given two $T$-coalgebras $f : a \rightarrow T(a)$ and $g : b \rightarrow T(b)$, a bisimulation between them is a subobject $r$ of the categorical product, together with a span $h : r \rightarrow a, k : r \rightarrow b$ for which there exists a coalgebra $l : r \rightarrow T(r)$ making $h$ and $k$ coalgebra morphisms from $l$ to $f$ and $g$, respectively. The greatest bisimulation, if it exists, is called bisimilarity.
Coalg(T) does not necessarily have a final object. If it does, the greatest bisimulation is obtained from the morphisms into the final coalgebra. An important case is when C is locally presentable and T is accessible. Here we recall the needed definitions and the theorem that we use, directly taken from [10]. See [40] for a detailed introduction.

**Definition 2.14 (locally presentable category)** Let λ denote an infinite regular cardinal. An object o in a category C is λ-presentable if its homset functor hom(o, −) preserves λ-filtered colimits. A category C is locally λ-presentable if it is cocomplete and there is a set A of λ-presentable objects such that every object is a λ-filtered colimit of objects from A. C is locally presentable if it is locally λ-presentable for some λ.

**Definition 2.15 (accessible functor)** Let C₁ and C₂ be locally λ-presentable categories. A functor T : C₁ → C₂ is λ-accessible if it preserves λ-filtered colimits, and it is accessible if it is λ-accessible for some λ.

Accessible categories generalize algebraic lattices [3]. Examples are the category Set, varieties of finitary many-sorted algebras, and the Schanuel topos. By equivalence, also named sets and permutation algebras are locally presentable. Roughly, accessible functors are determined by their action on objects whose cardinality is smaller than λ. Examples include polynominal functors, and any power set bounded by a cardinal λ. Accessibility is preserved by composition of functors.

The theory of coalgebras induces a generalisation of partition refinement known as iteration along the terminal sequence, used in [17] to minimise finite-control π-calculus agents. We do not present this algorithm here because it is not required, but we summarise the results of [40] and related works as follows.

**Theorem 2.16** Any accessible endofunctor on a locally presentable category admits a final coalgebra. If the functor preserves monos, then the terminal sequence converges to the final coalgebra.

Convergence to the final coalgebra (in a possibly infinite number of steps) does not guarantee termination in a finite number of steps, however it ensures that, if the algorithm terminates, it yields the same model for bisimilar systems, that is, it returns canonical representatives of classes of bisimilar states.

### 2.4 Multi-colimits

In [15] a weakened form of categorical limit, called multi-limit, is defined, where the limiting cocone is not unique. In this work, we are just interested in the special case of multi-coproducts, whose definition is given below.
Definition 2.17 (multi-coproduct) Given a diagram $D$ of objects, the multi-coproduct of $D$ is a set $\text{MCP}(D)$ of cocones over $D$ such that for all cocones $L' = \langle f'_i : o_i \to o' \rangle_{o_i \in |D|}$ over $D$ there is a unique $L = \langle f_i : o_i \to o \rangle_{o_i \in |D|} \in \text{MCP}(D)$ and a unique $u : o \to o'$ making $L \cup L' \cup \{u\}$ commute.

We use multi-coproducts as minimal canonical representatives, w.r.t. the order defined by existence of unique mediating morphisms, of sets of cocones that are equivalent, in the sense that they map things in the same way into an intermediate object, whose identity does not matter.

3 Behavioural Functors for Permutation Algebras

We describe how commonly used functors for the specification of the semantics of process calculi are defined in the category of permutation algebras. In §3.1 from a simple theory morphism, we define the name abstraction functor, study its properties, and give comparison with the definition of abstraction in [22]. §3.2 deals with the product, coproduct and power set functors. In §3.3 we show that the abstract syntax of $\lambda$-calculus using De Bruijn indexes is obtained as an initial algebra for a specific functor in $\text{FSAlg}^\pi$ involving our definition of name abstraction. §3.4 shows that the semantics of the $\pi$-calculus in [33] is actually making use of the functor we define here.

3.1 Name Abstraction

We first define a theory morphism called the right shift.

Definition 3.1 (right shift) The theory morphism $(-)^{+1} : \text{Autf} \to \text{Autf}$ is defined as $\pi^{+1}(i) = 0$ if $i = 0$ and $\pi^{+1}(i) = \pi(i - 1) + 1$ otherwise.

Definition 3.2 (functor $\delta$) The name abstraction functor $\delta : \text{Alg}^\pi \to \text{Alg}^\pi$ is obtained from the right shift theory morphism by Def. 2.17. It acts on objects as $\delta(\langle A, \{\pi_A\}\rangle) = \langle A, \{\pi_A^{+1}\}\rangle$ and on arrows as $\delta(f) = f$.

Notice that the action of a permutation in $\delta(A)$ cannot touch the name 0, due to the definition of $\pi^{+1}$. We now study the support, symmetry and orbits of elements of finitely supported permutation algebras obtained using $\delta$.

Theorem 3.3 The support and symmetry of elements of $\delta(A)$ are obtained as $\text{supp}_{\delta(A)}(a) = \{i - 1 \mid i \in \text{supp}_A(a) \setminus \{0\}\}$ and $\text{G}_{\delta(A)}(a) = \{\pi \mid \pi^{+1} \in \text{G}_A(a)\}$.

The above theorem proves that $\delta$ restricts from $\text{Alg}^\pi$ to $\text{FSAlg}^\pi$. Roughly, in $\delta(A)$, 0 is removed from the support of each element, and becomes fresh in
\(\delta(A)\). No observation can be made about 0, but it is still relevant in 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 \(\delta(A)\): \(\mathcal{G}(A)(a)\) is the subgroup of \(\mathcal{G}(a)\) that fixes 0, shifted by one name. Roughly, 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 \(\delta(A)\). Below, the finite set \(S\) is meant to be used as the support of an element of a permutation algebra, hence by Remark 3.5, we define these permutations just on \(S\).

**Definition 3.4 (binding permutations)** For \(S \in \mathcal{P}_{\text{fin}}(\omega)\), define a permutation \(\pi^{\text{old}}(S)\) such that \(\pi^{\text{old}}(S)(i) = i + 1\) for \(i \in S\), and \(|S|\) permutations \(\pi^{\text{hid}}(S, n)\) for \(n \in S\), such that \(\pi^{\text{hid}}(S, n)(i) = 0\) if \(i = n\) and \(\pi^{\text{hid}}(S, n)(i) = i + 1\) if \(i \in S \setminus \{n\}\).

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

**Definition 3.5 (metalanguage of binding)** The old element \(\text{old}_A(a)\) and the \(i\)th hidden element \(\text{hid}_A(a)\) of \(a \in A\) are defined as \(\text{old}_A(a) = \pi^{\text{old}}_{a, \text{supp}(a)}(a)\) and \(\text{hid}_A(a) = \pi^{\text{hid}}_{a, \text{supp}(a)}(a)\).

Old and hidden elements, by injectivity of permutations, form a partition of \(\delta(A)\). It is easily seen that \(\text{old}\) is an equivariant function from \(A\) to \(\delta(A)\) that preserves and reflects the symmetry and support of elements. The crucial property of \(\text{hid}\) is to send name \(i\) to 0, hence we have (by Thm. 3.3), \(\text{supp}_{\delta(A)}(\text{hid}_A(a)) = \text{supp}_A(a) \setminus \{i\}\): 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 in §3.4, such an operation is fundamental to define coalgebras for \(\delta\), allowing these to allocate fresh names along transitions. We can now define a semantic binding operation.

**Definition 3.6 (binder)** For each permutation algebra \(A\), \(a \in A\) and \(i \in \omega\), the binder \(\lambda_{A,i}.a\) is defined as \(\text{old}_A(a)\) if \(i \notin \text{supp}_A(a)\), \(\text{hid}_A(a)\) otherwise.

**Example 3.7** The product \(\omega \times \omega\) is a nominal set whose permutation action is \(\pi(\langle x, y \rangle) = (\pi(x), \pi(y))\) with trivial symmetry and \(\text{supp}(\langle x, y \rangle) = \{x, y\}\) (see §3.2). The carrier of \(\delta(\omega \times \omega)\) is the same, but the support and symmetry of elements is different. For \(i \neq x \neq 0\), we have \(\langle 0, x \rangle = \text{hid}(\langle i, x - 1 \rangle)\) and \(\text{supp}(\langle 0, x \rangle) = \{x - 1\}\), similarly \(\langle x, 0 \rangle = \text{hid}(\langle x - 1, i \rangle)\). The name 0 is a bound name, not appearing in the support of an element. For \(x, y \neq 0\) we have \(\langle x, y \rangle = \text{old}(\langle x - 1, y - 1 \rangle)\), and \(\text{supp}(\langle x, y \rangle) = \{x - 1, y - 1\}\).
Using this basic meta-language, we can more easily express the relationship between orbits of $\delta(\mathcal{A})$ and orbits of $\mathcal{A}$.

**Theorem 3.8** For $a \in A$, let $H^a = \{\text{old}_{\mathcal{A}}(a)\} \cup \{\text{hid}_{\mathcal{A}}(a) | i \in \text{supp}_{\mathcal{A}}(a)/\equiv\}$, where $i \equiv j \iff \exists \pi \in G_{\mathcal{A}}(a). \pi(i) = j$. Let $A^\mathcal{A}_a$ be a set of canonical representatives of orbits of $A$. A set $A^\mathcal{A}_\delta(\mathcal{A})$ of canonical representatives of orbits of $\delta(\mathcal{A})$ is obtained as $A^\mathcal{A}_\delta(\mathcal{A}) = \bigcup_{a \in A^\mathcal{A}_a} H^a$.

For each orbit in $A$, represented by $a^\mathcal{A}_o$, there is a corresponding orbit in $\delta(A)$ without any hidden name, plus as many orbits in $\delta(A)$ as the possible abstractions of names in $\text{supp}_{\mathcal{A}}(a^\mathcal{A}_o)$, modulo its symmetry. Roughly, there are as many ways to hide a name in $a^\mathcal{A}_o$ as the 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 $G_{\mathcal{A}}(a^\mathcal{A}_o)$.

**Example 3.9** It is interesting to apply the result of Thm. 3.8 to symmetries obtained by round shifts. 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 $P(1, 2, 3) = 12 + 23 + 31$. Its symmetry is $\{\text{id}, \sigma, \sigma \circ \sigma\}$, 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 $\{\text{id}\}$: 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 $\rho_{Pi}(P_1) = (\nu 1)(13 + 32 + 21)$ which is not structurally equivalent to $P_1$.

Finally, we compare the notions we have just obtained to the nominal set of abstractions of [22]. There, abstraction $[i]a$ for an element $a$ and a name $i$ is defined as the equivalence class obtained by swapping $i$ with a name $j$ in the pair $(i, a)$, 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 De Bruijn indexes [14]. By Thm. 2.9 objects obtained from the two constructions are isomorphic. This puts De Bruijn indexes and $\alpha$-conversion at the same level of abstraction.

### 3.2 Product, Coproduct and Power Set

It is well known that in a presheaf category $\text{Set}^C$ over a small category $C$, pointwise (co)limits are (co)limits. If $C$ has only one object, then this amounts to say that limits and colimits in $\text{Set}$ also are limits and colimits in $\text{Set}^C$. For the product and coproduct in $\text{Set}^{Autf} = \text{Alg}^\pi$, we have the following definitions.

**Definition 3.10** ((co)product) The product of $\mathcal{A}$ and $\mathcal{B}$ is $\langle A \times B, \{(\pi_A; \pi_B)\} \rangle$.
whose permutation action is the pairing of $\pi_A$ and $\pi_B$. Their coproduct is $\langle A + B, \{[\pi_A; \pi_B]\} \rangle$ whose permutation action is the copairing of $\pi_A$ and $\pi_B$.

The pairing and copairing of two permutation algebra morphisms (giving rise to the action of the functors on arrows) are just the corresponding set-theoretical notions. Algebraic properties of the coproduct (i.e. symmetry and orbits of elements) are trivial. For the product, we show that it restricts to an endofunctor in $\text{FSAlg}^\pi$.

**Theorem 3.11** The symmetry of $\langle a, b \rangle$ in $\mathcal{A} \times \mathcal{B}$ is given by $G_{\mathcal{A} \times \mathcal{B}}(\langle a, b \rangle) = G_A \cap G_B$. The support is $\text{supp}_{\mathcal{A} \times \mathcal{B}} = \text{supp}_A(a) \cup \text{supp}_B(b)$.

Now we give the definition of the covariant power set over permutation algebras. In the non finitely-supported case the definitions just lift from $\text{Set}$.

**Definition 3.12** The power set of $\langle A, \{\pi_A\} \rangle$ acts on objects as $\langle P(A), \{\pi_{P(A)}\} \rangle$ where $\pi_{P(A)}(p) = \{\pi_A(a) \mid a \in p\}$, and on arrows as $P(f)(p) = \{f(a) \mid a \in p\}$.

The countable power set can be defined by a size restriction. However, it does not restrict to an endofunctor in $\text{FSAlg}^\pi$. For example, the set of odd numbers is not finitely supported as an element of $P_{\text{cnt}}(\omega)$ [21]. This motivates the definition of a finitely supported power set, that we just denote with $P_{\text{cnt}}$ as it is “the” notion of countable power set that we use.

**Definition 3.13 (countable power set)** The finitely supported power set $P_{\text{cnt}} : \text{FSAlg}^\pi \rightarrow \text{FSAlg}^\pi$ is the restriction of the countable power set in $\text{Alg}^\pi$ to all the finitely supported subsets.

A set $p \subseteq A$ is finitely supported when, for a cofinite set $\tilde{S} = \omega \setminus S$, with $S$ finite, for all $\rho \in \text{fix}(S)$, we have $\rho_{\pi_A}(p) = p$. This happens when each $a \in p$ is supported by $S$, and also when it is supported by a set $\text{supp}_A(a)$ larger than $S$, if $p$ is closed w.r.t. all permutations mapping $\text{supp}_A(a) \setminus S$ into $\omega \setminus S$. As an example, let $\sigma^{(i,j)}$ be a permutation swapping $i$ with $j$, $a \in A$, $i \in \text{supp}_A(a)$. The set $r = \{\sigma^{(i,j)}(a) \mid j \in (\omega \setminus \text{supp}_A(a)) \cup \{i\}\}$ is supported by $\text{supp}_A(a) \setminus \{i\}$: for $\rho \in \text{fix}(\text{supp}_A(a) \setminus \{i\})$, we have $\{\rho_A(a) \mid a \in r\} = r$, i.e. the name $i$ “disappears” from the support of $r$. The set $r$ is infinite but “well behaved”; this is used in §6.5 to finitely represent a subfunctor of $P_{\text{cnt}}$.

3.3 Example: De Bruijn indexes

We illustrate how De Bruijn indexes are obtained as an initial algebra of a functor using $\delta$. We denote the elements of a coproduct $X_0 + X_1 + \ldots + X_n$ with $\{\langle i, x \rangle \mid i \in \{0, \ldots, n\} \land x \in X_i\}$. 


The syntax of the \( \lambda \)-calculus terms is defined by
\[
L ::= \lambda x.L \mid LL \mid x \quad \text{for } x \in \omega.
\]
Instead of introducing the notion of \( \alpha \)-equivalence for terms, we can define the syntax as the initial algebra \( \Lambda \) of the functor
\[
T(X) = \delta(X) + X \times X + \omega,
\]
were \( \omega \) is equipped with the permutation action \( \pi_\omega(i) = \pi(i) \). \( \Lambda \) is an arrow in \( \text{FSAlg}^\pi \) of type \( T(X) \to X \). An action of the permutation \( \pi_\Lambda \) on \( \Lambda \) is defined by initiality, thus introducing support, symmetry and orbits of terms. An interpretation \( f \) of \( \lambda \)-terms as elements of \( \Lambda \) is given in Fig. 1.

The most important case is the usage of \( \text{hid}_\Lambda^x \) in \( f(\lambda x.l) \): it shifts all the names of \( l \) by one, and assigns the name \( 0 \) to the bound name. All the \( \alpha \)-equivalent terms of the form \( \lambda y.l[y/x] \) are translated into the same element of \( \Lambda \), where \( 0 \) is not observable. The support of a term is therefore the set of its free names.

\[\text{Fig. 1. The abstract syntax of the } \lambda \text{-calculus in } \text{FSAlg}^\pi\]

\[
\begin{align*}
\text{rule 1:} & \quad \frac{X \xrightarrow{\rho(X)} X'}{\rho(l) \xrightarrow{\rho(l)} \rho(X')} \quad \text{for } l \in \mathcal{L}' \\
\text{rule 2:} & \quad \frac{X \xrightarrow{\rho(X)} X'}{\rho(\text{bout}(x)) \xrightarrow{\rho(\text{bout}(x))} \rho^+(X')}
\end{align*}
\]

\[\text{Fig. 2. The transition specification for the } \pi \text{-calculus}\]

3.4 \textit{Example: the } \pi \text{-calculus}

In [33], the early semantics of the \( \pi \)-calculus is given in the form of \textit{Structural Operational Semantics} rules, implicitly defining a functor for the \( \pi \)-calculus. Results in [13] ensure a lifting of the semantics from the category \( \text{Set} \) to the category \( \text{Alg}_\pi \) obtaining a bialgebra where the algebraic operations and axioms are those of the permutation signature. The lifting is subject to the fact that the coalgebra in \( \text{Set} \) respects the transition specification \( \Delta_\pi \) reported in Fig. 2. In the following, we use 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 \( \{\text{tau}, \text{in}(x,y), \text{out}(x,y)\} \) and \( \mathcal{L}'' \) with carrier \( \{\text{bout}(x)\} \).

\textbf{Theorem 3.14} Let \( T(X) = \mathcal{P}_{\text{cnt}}(\mathcal{L}' \times X + \mathcal{L}'' \times \delta(X)) \). A transition function in \( \text{FSAlg}^\pi \) is a coalgebra for \( T \) if and only if it respects \( \Delta_\pi \).

An arrow \( f : Pi \to T(Pi) \) representing the semantics of the \( \pi \)-calculus is described by the rules in Fig. 3 where we do not use additional labels to discriminate the two components of the coproduct, since the labels in \( \mathcal{L}' \) and \( \mathcal{L}'' \) are disjoint. As in [33], we employ the ordinary transition system in \( \text{Set} \).
\[
p \xrightarrow{\tau} p' \\
\langle \text{tau}, p' \rangle \in f(p)
\]
\[
p \xrightarrow{x} p' \\
\langle \text{in}(x, y), p' \rangle \in f(p)
\]
\[
p \xrightarrow{xy} p' \\
\langle \text{out}(x, y), p' \rangle \in f(p)
\]
\[
p \xrightarrow{\bar{x}y} p' \\
\langle \text{bout}(x), \text{hid}_{\pi_i}(p') \rangle \in f(p)
\]

Fig. 3. The coalgebra for the π-calculus

in the premises of the rules. The permutation \(\sigma\), applied to \(p'\) in the rule for bound output in \([33]\), gives exactly \(\text{hid}_{\pi_i}(p')\), hence the coalgebra provided there is the same as the one we introduced.

**Remark 3.15** The functor \(T(X) = P_{\text{cnt}}(\mathcal{L} \times \delta(X))\), with \(\mathcal{L} = \mathcal{L}' + \mathcal{L}''\) can as well be used, as \(X\) is embedded in \(\delta(X)\). Notice how close it is to the functor \(P(\mathcal{L} \times -)\), whose coalgebras in \(\text{Set}\) are labelled transition systems. A binding functor \(\delta\) is essentially the only addition to ordinary LTSs which is needed to represent the early semantics of the π-calculus. In other words, the abstraction functor allows one to enrich in a natural way ordinary labelled transition systems with dynamic allocation of names.

### 4 Named Sets and Local Names

Named sets are defined as sets whose elements have an associated set of names, and a symmetry over them. As names are local, each morphism is made up of a function \(h\) between the underlying sets, and an injective relabelling linking the names of \(h(q)\) to the names of \(q\), called the history of names. Here we provide a more structured notation for named sets, taking into account \([39,19]\). In particular, we employ an intermediate category representing injective name mappings quotiented by permutation groups. We also recast under the new definitions the equivalence between nominal sets and named sets of \([23]\).

For readability, we extend the notation for function composition to sets of functions having a common domain and codomain, by composing all the possible pairs; \(\text{dom}(F)\) and \(\text{cod}(F)\) are extended accordingly.

**Definition 4.1 (Symset)** For \(S \in P_{\text{fin}}(\omega)\), let \(\text{Grp}_S\) be the set of permutation groups over \(S\). The set of objects of \(\text{Symset}\) is \(\bigcup_{S \in P_{\text{fin}}(\omega)} \text{Grp}_S\). An arrow from \(\Phi_1\) to \(\Phi_2\) is a set of functions \(i \circ \Phi_1\) such that \(i : \text{dom}(\Phi_1) \rightarrow \text{dom}(\Phi_2)\) and \(\Phi_2 \circ i \subseteq i \circ \Phi_1\). We define \(\text{id}_\Phi = \text{id} \circ \Phi = \Phi\) and \(G \circ F = \{g \circ f \mid g \in G \wedge f \in F\}\).

Objects of \(\text{Symset}\) are groups of permutations over finite subsets of the natural numbers. An arrow represents an injective relabelling \(i\) from \(\text{dom}(\Phi_1)\) to \(\text{dom}(\Phi_2)\), quotiented by composition with the source group, that is, the
We call $S$ a set, and composition: for $G = i_2 \circ \Phi_2$ and $F = i_1 \circ \Phi_1$, we have $G \circ F = (i_2 \circ i_1) \circ \Phi_1$. The category $I$ of finite sets and injections is a full subcategory of $\text{Symset}$.

**Definition 4.2 (named set)** A named set is a pair $N = (Q_N, S_N)$, with $Q_N$ a set, and $S_N$ a function from $Q_N$ to the set $\text{Symset}$ of objects of $\text{Symset}$. We call $\|q\|_N = \text{dom}(S_N(q))$ the set of names of $q$. A named function from $N$ to $M$ is a pair $F = (h_F, \Sigma_F)$ where $h_F : Q_N \rightarrow Q_M$ and $\Sigma_F$ is a function giving for each $q \in Q_N$ a morphism $\Sigma_F(q) : S_M(h_F(q)) \rightarrow S_N(q)$. We define $\text{id}_N = (\text{id}_{Q_N}, \lambda q. \text{id}_{S_N(q)})$, and $G \circ F = (h_G \circ h_F, \lambda q. \Sigma_F(q) \circ \Sigma_G(h_F(q)))$.

A named set is a set of elements, each one having an associated object of $\text{Symset}$ that represents its finite set of names and a symmetry over them. A named function $F$ has two components: $h_F$, which is an ordinary function over $Q_N$, and $\Sigma_F$, giving for each $q \in Q_N$ an arrow of $\text{Symset}$, denoting a backward mapping from the symmetry and names of $h_F(q)$ to those of $q$.

**Remark 4.3** Even though it is not required in this work, it is immediate from the definition of a named set that the category $\text{NSet}$ is the free coproduct completion of the category $\text{Symset}^\text{op}$, similarly to the characterisation of $\text{Symset}$.

A mathematical description of stateful systems with name allocation can be given using History-Dependent automata (HD-automata).

**Definition 4.4 (HD-automaton)** An HD-automaton is a coalgebra in $\text{NSet}$, i.e. a named (transition) function $F : N \rightarrow T(N)$, for $N$ a named set (of states) and $T$ some endofunctor over $\text{NSet}$.

We now recall the equivalence result proved as proposition 29 in [23].

**Theorem 4.5** The categories $\text{NSet}$ and $\text{FSAlg}^\pi$ are equivalent by the functors $F : \text{FSAlg}^\pi \rightarrow \text{NSet}$ and $G : \text{NSet} \rightarrow \text{FSAlg}^\pi$.

**Definition 4.6** $F$ maps $\langle A, \{\pi_A\} \rangle$ to $\langle A^o, S_N \rangle$, with $S_N(a^o) = \mathcal{G}_A(a^o|_{\mathcal{supp}_A(a^o)})$, and $f : A \rightarrow B$ into $\langle h_f, \Sigma_f \rangle$ with $h_f(a^o) = f(a^o)^o$, $\Sigma_f(a^o) = \rho \circ S_N(f(a^o)^o)$, and $\rho$ such that $\rho_B(f(a^o)^o) = f(a^o)$.

Orbits “collapse” into canonical representative. All elements differing for an injective renaming are identified, discarding the global meaning of names.

**Example 4.7** Since $\omega$ has just one orbit, a name and trivial symmetry in the support of each element, the corresponding named set is $\langle \{0\}, \lambda x. \{\text{id}_{\{0\}}\} \rangle$.

**Definition 4.8** $G$ maps a named set $N$ to $\langle A_N, \{\pi_N\} \rangle$, where $A_N = \{\langle q, \rho \circ S_N(q) \rangle : q \in Q_N, \rho : \|q\|_N \rightarrow \omega\}$, and $\pi_N(\langle q, I \rangle) = \langle q, \pi \circ I \rangle$. $G$ maps $F :
\( N \to M \) to \( G(F)((q, \rho \circ S_N(q))) = \langle h_F(q), \rho \circ S_N(q) \circ \Sigma_F(q) \rangle \).

5 Name Abstraction in Named Sets

Here we define the abstraction functor in named sets, called \( H \) (for “hiding”). For an element \( q \) of a named set, we define an equivalence \( i \equiv_q j \iff \exists \pi \in S_N(q). \pi(i) = j \). Its canonical representatives are denoted with \([i]_q \) for \( i \in ||q|| \).

**Definition 5.1** \( H \) maps a named set \( N \) to \( (Q, S) \), where \( Q = Q_N \cup \{(q, i) \mid q \in Q_N, i \in (||q||_N)/\equiv_q \} \) and \( S(q) = S_N(q), S((q, i)) = gfix (S_N(q), i) \).

Notice that the union in the definition of \( Q \) is disjoint. The elements of \( Q_N \) represent the orbits of the old elements of Def. 3.5 and each pair \((q, i)\) represents the orbit of the \( i^{th} \) hidden element of \( q \). From Thm. 3.3 the possible values for \( i \) are quotiented with \( \equiv_q \). Informally, \( i \) marks the \( i^{th} \) name of \( q \) as hidden. The symmetry of elements of the form \((q, i)\) is the subgroup of the symmetry of \( q \) that fixes \( i \) (which we denote with \( gfix (S_N(q), i) \) according to Thm. 3.3. This group is restricted to exclude \( i \) from the support.

**Definition 5.2** \( H \) maps \( F : N \to M \) to \( \langle h, \Sigma \rangle \), where \( h(q) = h_F(q), h((q, i)) = \langle h_F(q), [j] \rangle \) if \( \exists \sigma \in \Sigma_F(q). \sigma(j) = i \), \( h_F(q) \) otherwise, \( \Sigma(q) = \Sigma_F(q), \Sigma((q, i)) = \{\sigma_{|\text{dom}(\sigma) \setminus \{i\}} \mid \sigma(j) = i \wedge \sigma \in \Sigma_F(q)\} \) if \( h((q, i)) = \langle h_F(q), j \rangle \), \( \Sigma_F(q) \) otherwise.

A name \( j \) is present (but hidden) in the destination iff it is mapped by some injection in \( \Sigma_F(q) \) into \( i \), that is, iff \( i \) is a name of \( h_F(q) \), according to the history of names \( \Sigma_F(q) \). Notice that \( h \) must respect the equivalence relation on hidden names. This comes from Def. 4.1 for each \( \sigma \in \Sigma_F(q) \) we have \( \Sigma_F(q) = \sigma \circ S_M(h_F(q)) \). 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 \( S_M(h_F(q)) \), hence \( j \) and \( j' \) are equivalent. When a hidden name is still present in the destination, \( \Sigma(q) \) is the subset of \( \Sigma_F(q) \) that sends \( j \) into \( i \), which is then restricted to \( \text{dom}(\sigma) \setminus \{j\} \), so that \( j \) is not mapped at all.

Finally, we prove a correctness result for our definition, with respect to the one in \( \text{FSAAlg}^\pi \) (more details in §7). The functor \( G \) is from Def. 4.8.

**Theorem 5.3** \( H \) is a functor. The two functors \( G \circ H \) and \( \delta \circ G \) are isomorphic, i.e. there exists a natural isomorphism \( \iota : G \circ H \to \delta \circ G \).

In the action of \( H \) on arrows, when a hidden name \( i \) is discarded along a morphism, the resulting element is just \( q \). This introduces garbage collection in the model, as old states can be reused. We now show three examples in the \( \pi \)-calculus, aimed at explaining what we mean with locality of names. In particular, we see 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.

Consider the agent $P(1) = (\nu x)\overline{1}.P(1)$. Even though it has no memory of
the past, the permutation algebra semantics of the system reaches all the (in-
finite) elements in the orbit of $P(1)$. Fig. 4 depicts a sketch of the semantics
in $\mathbf{FSAlg^π}$, compared to the one in $\mathbf{NSet}$, 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 $h_tr(P(1)) = \{P(1)\}$, $\Sigma_tr(P(1)) = \{id_{\{1\}}\}$.

Let $R(1) = (\nu x)\overline{1}.R(x)$. Consider a presheaf semantics for the $\pi$-calculus as
in [20]. On the left of each state, we draw in Fig. 6 the least stage (object of the
base category) in which the state is found (the categorical support in [23]), and
the stage in which the coalgebra is applied to it. Recall that, given a presheaf $T$, the functor $\delta : \mathbf{Set}^I \rightarrow \mathbf{Set}^I$ is defined on objects as $\delta(T)(X) = T(X \oplus 1)$. To
distinguish the different instances of $* \in 1$, in the successive applications of
the coproduct, we denote these with $*'$, $*''$ and so on. In Fig. 7, we depict the
HD-automata semantics of $R(1)$, made of two states. The transition function is
$h_tr(R(1)) = \{(R(1), 1)\}$, thus hiding name 1, and $h_tr((R(1), 1)) = \{(R(1), 1)\}$. We have $\Sigma_tr(R(1)) = \Sigma_tr((R(1), 1)) = \{\emptyset\}$, the empty name mapping (the
support of the destination is empty).

Now let us define $T(x, y) = (\nu z)(\nu w) (\exists z.T(z, y) + \exists w.T(x, w))$. This agent has
two names, and its symmetry is non-trivial: the swap $\sigma^{(x,y)}$ gives a structurally
congruent agent. Since there is recursion with name allocation, the global-
names semantics are infinite, but the HD-automata one still performs two
initial steps, similarly to $R$ (but executed in arbitrary order), and then contains
two looping transitions over $T(x, y)$, each one allocating a name and discarding
another one. Roughly, we may think of the local names of a state as a memory
that contains the names that are observable in the future transitions, hence
have to be recorded in the state. $P$ and $R$ just need to store one name, while
$T$ needs to store two names, and this is observed in its support. In this light,
the functor $\delta$ provides a memory allocation primitive, and garbage collection
is implicit in the model.

6 Local Names in Global Environments

Here we show how, starting from entities having local names, one can build
“larger” entities, that is, we introduce the product and power set constructions,
and, for completeness, the coproduct. The product and power set constructions
make use of the multi-coproduct in $\mathbf{Symset}$, that we define as a first step.
Fig. 4. Infinite states in $\text{FSAlg}^\pi$

Fig. 5. Finite states in $\text{NSet}$

Fig. 6. Infinite states in $\text{Set}^I$

Fig. 7. Finite states in $\text{NSet}$
6.1 Multi-Coproducts in Symset

Consider a cospan \( \Phi_1 \xrightarrow{f} \Phi \xleftarrow{g} \Phi_2 \) in Symset. This may be interpreted as a symmetric name correspondence between \( \Phi_1 \) and \( \Phi_2 \), allowing some names to be identified, and others to be kept distinct. The middle object is only used as a representation of the common and distinguished names of the two objects. Therefore, we are not interested in names of \( \Phi \) that are not in the image of \( f \) or \( g \), and in distinguishing two isomorphic cospans, as the obtained name mapping is the same. The multi-coproduct models this situation.

\[ \text{Theorem 6.1} \quad \text{The multi-coproduct of } \langle \Phi_1, \Phi_2 \rangle \text{ in Symset is given by the set of cospans } \langle \text{in}_1; \Phi_1 \rightarrow \Phi, \text{in}_2; \Phi_2 \rightarrow \Phi \rangle \text{ such that } \text{dom}(\Phi) = \text{Im}(\text{in}_1) \cup \text{Im}(\text{in}_2) \text{ and } \Phi \text{ is the greatest permutation group over dom}(\Phi) \text{ such that in}_1 \text{ and in}_2 \text{ respect the conditions of Def. 4.1, quotiented by cospan isomorphism.} \]

Informally, names in \( \Phi \) are meaningful since they are in the image of at least one \( \text{in}_i \). Maximality of \( \Phi \) ensures existence of the unique mediating morphism.

6.2 Categorical Product of Named Sets

The categorical product generalises the set-theoretical notion of relation. In named sets the product exemplifies locality of names, i.e. that a binding of names has to be established when two elements are related.

\[ \text{Theorem 6.2} \quad \text{The product } N \times M \text{ is } \langle Q, \mathcal{S} \rangle, \text{ where } Q \text{ contains the tuples } \langle n, m, (\text{in}_1, \text{in}_2) \rangle \text{ where } n \in Q_N, m \in Q_M, \langle \text{in}_1, \text{in}_2 \rangle \in \text{MCP}(\langle \mathcal{S}_N(n), \mathcal{S}_M(m) \rangle), \text{ and } \mathcal{S}(\langle n, m, (\text{in}_1, \text{in}_2) \rangle) = \text{cod}(\text{in}_1) = \text{cod}(\text{in}_2). \text{ The projections } \pi_1 = \langle h_1, \Sigma_1 \rangle \text{ and } \pi_2 = \langle h_2, \Sigma_2 \rangle \text{ are defined as } h_1(t) = n, \Sigma_1(t) = \text{in}_1, h_2(t) = m, \Sigma_2(t) = \text{in}_2. \]

The underlying set \( Q \) is made of triples of an element of \( Q_N \), an element of \( Q_M \), and an element of a multi-coproduct in Symset. The common target of \( \text{in}_1 \) and \( \text{in}_2 \) represents the symmetry (and the names) of an element of the categorical product. An intuition can be given starting from permutation algebras. For \( \langle A, \{ \pi_A \} \rangle \) and \( \langle B, \{ \pi_B \} \rangle \) permutation algebras, \( a \in A \) and \( b \in B \), having orbits \( \{ \pi_A(a^\pi) \mid \pi \in \text{Autf} \} \) and \( \{ \pi_B(b^\pi) \mid \pi \in \text{Autf} \} \), an element of the product can be any pair \( \langle \pi_A(a^\pi), \pi_B(b^\pi) \rangle \). The maps \( \pi' \) and \( \pi'' \) give rise to \( \text{in}_1 \) and \( \text{in}_2 \) through the equivalence between \( \text{FSAAlg}^\pi \) and \( \text{NSet} \).

\[ \text{Definition 6.3 (pairing) } \text{The pairing of } F : N \rightarrow N' \text{ and } G : M \rightarrow M' \text{ is } \langle h, \Sigma \rangle. \text{ Let } t = \langle n, m, (\text{in}_1, \text{in}_2) \rangle \text{ be an element of } Q. \text{ Let } \langle \text{in}_1', \text{in}_2' \rangle \in \text{MCP}(\langle \mathcal{S}_N(n), \mathcal{S}_M(m) \rangle) \text{ be the unique element of the multi-coproduct commuting with } \langle \text{in}_1 \circ \Sigma_F(n), \text{in}_2 \circ \Sigma_G(m) \rangle \text{ and } u \text{ the associated unique arrow. Then we define } h(t) = \langle h_F(n), h_G(m), \langle \text{in}_1', \text{in}_2' \rangle \rangle \text{, and } \Sigma(t) = u. \]
The underlying functions $h_F$ and $h_G$ are paired as in Set. The injections are also paired, and composed with the name mappings $\Sigma_F(n)$ and $\Sigma_F(m)$. This results in a cocone that is not guaranteed to belong to a multi-coproduct. The multi-coproduct is used to find a canonical form for it. The name mapping associated to the pairing is then the unique commuting arrow (see Fig. 8).

**Example 6.4** Continuing from Example 4.7, consider the product $\langle Q, S \rangle$ of the named set $\omega$ with itself. Let $\Phi_1$ and $\Phi_2$ denote respectively the groups $\{id_{\{0\}}\}$ and $\{id_{\{0,1\}}\}$. The multi-coproduct in Symset of $\Phi_1$ with itself is represented by the two cospans $\langle \{f\}, \{f\} \rangle$ and $\langle \{g\}, \{g'\} \rangle$ where $f = id_{\Phi_1}$, $g : \Phi_1 \rightarrow \Phi_2$ with $g(0) = 0$, and $g' : \Phi_1 \rightarrow \Phi_2$, with $g(0) = 1$. That is, two names in the product $\omega \times \omega$ are either distinct or equal. Notice that there can not be a cospan whose middle object is the group $\Phi_3 = \{id_{\{0,1\}}, \sigma^{(0,1)}\}$, where $\sigma^{(0,1)}$ is the swap of 0 and 1. A Symset arrow from $\Phi_1$ to $\Phi_3$ would be a singleton consisting of a function (either $f$ or $g$), violating the additional condition of Def. 4.7. $Q$ consists of the two tuples $q_1 = \langle 0, 0, \{f\}, \{f\} \rangle$ and $q_2 = \langle 0, 0, \{g\}, \{g'\} \rangle$. The symmetry is $S(q_1) = \Phi_1$ and $S(q_2) = \Phi_2$.

To characterise the product, we employ the multi-coproduct in Symset. Actually, existence of multi-limits of a given shape in a category $C$ is equivalent to existence of limits of the same shape in its free coproduct completion $\text{Fam}(C)$ (see e.g. [12], remark 5). This is promising in view of generalising the construction of named sets from permutation algebras to more structured signatures (or presheaves with richer index categories than I).

A real-world example of the idea of the product in the presence of local names can be given. Consider two isolated networks, with each node having a name, locally known to the other machines of its neighbourhood. When linking the two networks, a larger domain of names is obtained, and a binding machinery, playing the role of the two arrows $in_1$ and $in_2$ in Def. 6.2, is needed to establish how names of each system are bound to a name in the common domain.

### 6.3 The Coproduct

The coproduct has a fairly simple structure, just inherited from the set-theoretical definition. We denote with $[f; g]$ the copairing of $f$ and $g$. 

![Diagram](image-url)
Theorem 6.5 The coproduct of $N = \langle Q_N, \mathcal{S}_N \rangle$ and $M = \langle Q_M, \mathcal{S}_M \rangle$ is $\langle Q_N + Q_M, [\mathcal{S}_N; \mathcal{S}_M] \rangle$, and the injections are the set-theoretical ones, with identity name mappings. The copairing of $F$ and $G$ is $\langle \{h_F; h_G\}, [\Sigma_F; \Sigma_G] \rangle$.

6.4 Power Set

As in the case of the product, in the countable power set functor $\mathcal{P}_{cnt}$, a multi-coproduct is used to relate names of the elements of each subset. One might expect an element of $\mathcal{P}_{cnt}(N)$ to be a set $p$ of pairs $\langle q_i, in_i; \mathcal{S}_N(q_i) \rightarrow \phi \rangle_{i \in I}$, with $I$ finite or countable. A problem is to consider the implicit symmetry due to $p$ being a set, not a tuple. Symmetries can not grow along the $in_i$ as they are morphisms of $\text{Symset}$, thus $\phi$ cannot contain this symmetry, as it is the target of all the $in_i$; locally, each $q_i$ might even not have symmetries at all. Thus, differently from the categorical product, $\phi$ is not the symmetry of $p$. Instead, it should be completed with additional permutations. Moreover, to represent infinite sets, a finite intermediate object $\phi$ may not be sufficient. We first give up on the finite support requirement on $\phi$, and then “recover” it by selecting only those subsets that actually are finitely supported. Consider the category $\text{Symset}^{\omega}$ whose objects are groups of permutations, having a finite or countable domain. $\text{Symset}^{\omega}$ has multi-coproducts of countable diagrams, and $\text{Symset}$ is a full subcategory of it.

Definition 6.6 (finite support) An object $\phi \in |\text{Symset}^{\omega}|$ is finitely supported if $\Phi \in |\text{Symset}|$ or there exists a finite supporting set $S \in \mathcal{P}_{\text{fin}(\omega)}$ such that $\Phi|_{\text{dom}(\Phi) \setminus S}$ contains all the possible permutations over $\text{dom}(\Phi) \setminus S$. We let $\text{fin}(\Phi)$ denote $\Phi$ in the first case, $\Phi|_T$, where $T$ is the smallest finite supporting set, in the second case. Notice that $\text{fin}(\Phi) \in |\text{Symset}|$.

For $N$ named set, and $\Phi \in |\text{Symset}^{\omega}|$, let $P(N, \Phi)$ be the set of finite or countable sets of pairs $\langle q, in \rangle$ such that $q \in Q_N$ and $in$ is a $\text{Symset}^{\omega}$ arrow from $\mathcal{S}_N(q)$ to $\Phi$. Let $P(N)$ denote $\bigcup_{\Phi \in |\text{Symset}^{\omega}|} P(N, \Phi)$. The completion below represents the unordered nature of the power set.

Definition 6.7 (completion) The completion of $p \in P(N)$ is an object of $\text{Symset}^{\omega}$, defined as the greatest permutation group $p^c$ such that $p = \{\langle q_i, \rho \circ in_i \rangle \mid \rho \in p^c \land \langle q_i, in_i \rangle \in p\}$.

The definition of the countable power set follows. The action of the functor on arrows employs the multi-coproduct in a similar way to Def. 6.3.

Definition 6.8 (countable power set) The countable power set is defined as $\mathcal{P}_{cnt}(N) = \langle Q, \mathcal{S} \rangle$ where $Q$ is the greatest subset of $P(N)$ such that for each $p = \{\langle q_i, in_i \rangle \mid i \in I\} \in Q$, the cocone $\langle in_i \rangle_{i \in I}$ is in $\text{MCP}(\{\mathcal{S}_N(q_i)\}_{i \in I})$ and $p^c$ is finitely supported. The symmetry is defined as $\mathcal{S}(p) = \text{fin}(p^c)$. The
functor maps $F : N \to M$ to $\langle h, \Sigma \rangle$. Let $p = \{\langle q_i, \text{id} \rangle \mid i \in I \}$ $\in Q$. Let $\langle in'_i \rangle_{i \in I}$ and $u$ denote respectively the unique element of $\text{MCP}((S_M(h_F(q_i)))_{i \in I})$ commuting with $\langle in_i \circ \Sigma(q_i) \rangle_{i \in I}$ and the associated unique morphism. Then we have $h(p) = \{\langle h_F(q_i), \text{id} \rangle \mid i \in I \}$, and $\Sigma(p) = u \circ S(h(p))$.

Finally, we state an isomorphism with the corresponding notion in $\text{FSAlg}^\pi$. Below, the functor $G : \text{NSet} \to \text{FSAlg}^\pi$ comes from Def. 4.8.

**Theorem 6.9** Let $P^F_{\text{cnt}}$ denote $P_{\text{cnt}}$ in $\text{FSAlg}^\pi$ and $P^N_{\text{cnt}}$ denote the corresponding functor in $\text{NSet}$. There is a natural isomorphism $\iota : P^F_{\text{cnt}} \circ G \to G \circ P^N_{\text{cnt}}$. Let $p = \{\langle q_i, \rho_i \rangle \mid i \in I \}$, for some $I \subseteq \omega$, be an element of $P^F_{\text{cnt}} \circ G(N)$. Notice that each $\rho_i$ can be regarded as an arrow in $\text{Symset}^\omega$ from $S(q_i)$ to $\{\text{id}_\omega\}$. Let $\langle in_i \rangle_{i \in I}$ and $\rho$ be respectively the unique cocone and the unique arrow commuting with $\langle \rho_i \rangle_{i \in I}$, coming from the multi-coproduct. Let $p' = \{\langle q_i, \text{id} \rangle \mid i \in I \} \in P^N_{\text{cnt}}(N)$. We have $\iota_N(p) = \langle p', \rho \circ S(p') \rangle$.

### 6.5 Finitely Representable Power Set

In this section we deal with a subfunctor of the power set whose action can be represented in a finite way. The idea behind it comes from the so-called “normalisation of bundles” that was used in [17] to minimise the early $\pi$-calculus. The solution we propose is not specific for the $\pi$-calculus; it can be used for any programming language in which it is not decidable whether a variable is actually used or not. The definition of the functor is not tied to named sets, so we can present it in the category $\text{FSAlg}^\pi$ for clarity. Consider a non-deterministic choice quantified over all the possible names, be those fresh or free. The prototypical situation is modelling an input transition in a name-passing formalism.

A technique which is quite natural to tackle this problem is to define a finitely-branching transition system that contains the finite number of input transitions that receive an already known (i.e. non-fresh) name, plus a single transition that represents all the ones that receive a fresh name. However, the problem of redundant names makes this approach incorrect. Consider for example the two $\pi$-calculus agents $P(x,y) = x(z) . \bar{z}y.0 \parallel (\nu w)\bar{w}y.0$ and $Q(x) = x(z) . \bar{z}z.0$. The prefix $\bar{w}y$ in the definition of $P$ does not trigger any output transition, because it is immediately under the scope of a restriction over its subject. Thus, the two agents are bisimilar. However, if we base our choice of “interesting” input transitions only on the free names of a process, we get two finitely-branching systems that are not bisimilar, since $P(x,y)$ has a “spurious” free input transition labelled with $xy$ which is not present in $Q(x)$. Name $y$ is free in $P(x,y)$, but it is not observable in the semantics: it is redundant. Free names of a process are a syntactic notion. Instead, we
are interested in observable names, belonging to the realm of the semantics. Unfortunately, redundancy of names is not decidable in general.

The key idea of \[17\] is to change the action of the functor for the early \(\pi\)-calculus on arrows, from that of the finite power set to a normalising variant of it. The action of the modified functor \(T\) removes from the destination \(T(f)(p)\) of a coalgebra \(f\) applied to a process \(p\) all the free input transitions that are proved to receive a fresh name (thus, to be redundant) in one step. To recast this machinery in our coalgebraic framework, we first define a subfunctor \(P_{fr}\) of \(P_{cnt}\), whose action on objects may contain infinite subsets, that can be represented in a finite way. Then, we exhibit a finitely branching representation functor \(P\) naturally isomorphic to it, even if not definable just using our framework of accessible endofunctors, because of its action on arrows.

**Definition 6.10 (Closure)** In the following, given a permutation algebra \(A\), \(a \in A\) and \(n \in \omega\), we define the closure of \(a\) on \(n\) in \(A\), written \(clos_{A}(a,n)\), as the set \(\left\{ \pi_{A}(a) \mid \pi \in \text{fix}(\text{supp}_{A}(a) \setminus \{n\}) \right\}\).

Notice that if \(n \notin \text{supp}_{A}(a)\) then \(clos_{A}(a,n) = \{a\}\). Otherwise, \(clos_{A}(a,n)\) is a cofinite set obtained by swapping \(n\) in \(a\) with all the possible names that are not in the support of \(a\).

**Definition 6.11 (Finitely representable power set)** The action of \(P_{fr}\) on objects is \(P_{fr}(A) = \langle P, \{\pi_{P_{cnt}(A)}\} \rangle\), where \(P\) contains all the \(p \in P_{cnt}(A)\) for which there exists a finite set of indexes \(I \in P_{fin}(\omega)\), a tuple of names \(\langle n_{i} \rangle_{i \in I}\) and a tuple of elements \(\langle a_{i} \rangle_{i \in I}\), such that \(p = \bigcup_{i \in I} clos_{A}(a_{i},n_{i})\). The action of the functor on arrows is that of the power set.

The intuition behind the above definition is that each set \(p \in P_{fr}(A)\) is built using a finite number of elements, each of which may be replicated infinitely many times by swapping a single name with all the others. The closure operation is similar to a binding operation, since we have \(n \notin \text{supp}_{A}(a)\). The idea of the rest of this section is to show how \(clos_{A}(a,n)\) can be finitely represented in \(\delta(A)\) as \(\lambda_{A} n.a\), hence, the whole \(P_{fr}(A)\) can be represented using \(P_{fin}(\delta(A))\). An issue arises in doing so: consider an element of \(P_{fin}(\delta(A))\) containing both an element \(hid_{A}^{a}(a)\) representing a closure \(clos(a,n)\) where \(n \in \text{supp}(a)\), and the element \(old_{A}(a)\). Notice that \(a\) belongs to \(clos(a,n)\). Such a set is distinct from its “standard” representation that only contains \(hid_{A}^{a}(a)\). Indeed, this is the reason why the naïve approximation of the semantics of the \(\pi\)-calculus above does not work. To give a finite representation of \(P_{fr}\), we thus define a normalisation operation as follows.

**Definition 6.12 (Normalisation)** For \(p \in P_{fin}(\delta(A))\), we define the function \(\text{norm}_{A}(p) = p \setminus \{old_{A}(a) \mid \exists y \in \text{supp}_{A}(a).hid_{A}^{y}(a) \in p\}\).
Using this function, that removes from $p$ all the elements that are already represented by an element with a bound name, we can define the correct representation of the functor $P_{fr}$.

**Definition 6.13 (Finite representation)** The functor $P$ acts on objects as $P(A) = \langle P_A, \{\pi_{\text{fin}}(\delta(A))\} \rangle$ where $P_A = \{\text{norm}_A(p) \mid p \in P_{\text{fin}}(\delta(A))\}$ and on arrows as $P(f : A \to B) = \text{norm}_B \circ P_{\text{fin}}(\delta(f))$.

The functor $P$ is not a subfunctor of $P_{fr}$, since the action on arrows employs normalisation to eliminate redundant transitions. Roughly, the functor eliminates the transitions that can be proved redundant in one step. As a side effect, iteration along the terminal sequence eliminates all the redundant transitions. This implies that the algorithm may not terminate, as redundant transitions may not be decidable (e.g. the $\pi$-calculus). A correctness result is given below.

Here we use the binding operation of Def. 3.6.

**Theorem 6.14** The functors $P_{fr}$ and $P$ are naturally isomorphic. Let $p \in P_{fr}(A)$, and assume a canonical choice of $I$, $\langle a_i \rangle_{i \in I}$ and $\langle n_i \rangle_{i \in I}$ such that $p = \bigcup_{i \in I} \text{clos}_A(a_i, n_i)$. The natural isomorphism $\iota : P_{fr}(A) \to P(A)$ is then obtained as $\iota(p) = \{\lambda_A n_i.a_i \mid i \in I\}$.

7 A Final Coalgebra Theorem

The idea that we present here is to lift an equivalence of categories to the corresponding categories of coalgebras of a pair of equivalent functors.

**Definition 7.1 (equivalent functors)** Given two categories $A$ and $B$, equivalent via $F : A \to B$ and $G : B \to A$, two functors $T : A \to A$ and $S : B \to B$ are equivalent if there is a natural isomorphism $k : F \circ T \to S \circ F$.

As equivalences of categories preserve all limits and colimits, if $T$ and $S$ are equivalent, $T$ is accessible if and only if $S$ is. It is also easy to see that the two categories of coalgebras are equivalent. The functor $\hat{F} : \text{Coalg}(T) \to \text{Coalg}(S)$ sends each coalgebra $f : a \to T(a)$ into $k \circ F(f) : F(a) \to S(F(a))$. We can now state a final coalgebra theorem.

**Theorem 7.2** Functors composed of (co)products, $\delta$, $P_{fr}$ and $P_{cnt}$ are accessible and have a final coalgebra in $\text{NSet}$.

Therefore, one can define coalgebras in nominal sets, where specifications are very close to set-theoretical ones (see §3.4), and then obtain a semantics in named sets (see §8.1) by the constructions that we presented in this work.
\[
\begin{align*}
    p \xrightarrow{\tau} p' & \quad \langle \text{tau}, \text{old}_{P_i}(p') \rangle \in f(p) \\
    p \xrightarrow{xy} p' & \quad \langle \text{in}(x, y), \text{old}_{P_i}(p') \rangle \in f(p) \\
    p \xrightarrow{\bar{xy}} p' & \quad \langle \text{out}(x, y), \text{old}_{P_i}(p') \rangle \in f(p) \\
    p \xrightarrow{y} p' & \quad \langle \text{bout}(x), \text{hid}_{P_i}(p') \rangle \in f(p)
\end{align*}
\]

Fig. 9. The coalgebra \( f \) for \( \mathcal{P}_{\text{cnt}}(\mathcal{L} \times \delta(-)) \)

8 Examples

8.1 The \( \pi \)-calculus

It is now easy to define the semantics of the \( \pi \)-calculus as a coalgebra in \( \text{NSet} \) by translating the semantics of §3.4 using the functor \( F \) (Def. 4.6). Following Remark 3.15, we define a semantics in \( \mathcal{P}_{\text{cnt}}(\mathcal{L} \times \delta(N)) \), where \( \mathcal{L} = \{ \text{tau}, \text{in}(x, y), \text{out}(x, y), \text{bout}(x) \mid x, y \in \omega \} \) (Fig. 9).

The semantics is infinitely branching only because of early input transitions, thus it falls under the conditions of Def. 6.11. We represent the functor as \( T = \mathcal{P}(\mathcal{L} \times H(-)) \), whose action on objects is included in that of \( \mathcal{P}_{\text{fin}}(H(\mathcal{L} \times H(N))) \). The first \( H \) represents the early input transitions, while the second one is used to represent bound output transitions. Roughly, the input transitions having a bound name are “bound input” transitions. The norm function removes at each step the free input transitions that can be proved redundant in one step, i.e. the destination is equal, up-to swapping the received name with a fresh one, to that of a bound input transition.

The correctness of the algorithm of [17] has been proved correct in the finite case. Thm. 7.2 guarantees that \( T \) has a final coalgebra, hence, iteration along the terminal sequence converges in the finite case, and returns the same result. The two representations are isomorphic, since the final coalgebra is unique up to isomorphism. The definition of all the functors we present in this work gives the proof of correctness “for free”, avoiding all the technicalities of the rather complex underlying set of \( T(N) \). The obtained semantics is fully abstract: from [33], Thm. 29, and from §3.4, we know that co-algebraic bisimulation for the functor \( \mathcal{P}_{fr}(\mathcal{L} \times \delta(-)) \) coincides with early bisimulation. By the equivalence \( F : \text{FSAlg}^\pi \rightarrow \text{NSet} \) and of the categories of coalgebras of equivalent functors, we know that co-algebraic bisimulation in \( \text{NSet} \) for the functor \( T \) coincides with early bisimulation, when the latter is translated to a subobject of the product in \( \text{NSet} \) via \( F \). This proof can be reused for other nominal calculi whose semantics is expressed in \( \text{FSAlg}^\pi \) using the functors we have presented, getting a correct semantics of each calculus that exploits named sets, from a simpler description using global names and binders.
It is interesting to see how elements of $T(N)$ are defined in terms of those of $N$. For each named set $N$, the possible transitions are in $P(\mathcal{L} \times \mathcal{H}(N))$, hence in $\mathcal{P}_{\text{fin}}(\mathcal{H}(\mathcal{L} \times \mathcal{H}(N)))$. The named set of labels has the following elements: \{tau, in(x, y), in(x, x), out(x, y), out(x, x), bout(x)\}, for $x$ and $y$ a fixed pair of names in $\omega$ (notice that different representatives are needed depending on the subject and the object of an action being equal or different). The symmetry of each label is trivial. Elements of the underlying set $Q$ of $T(N)$ are sets of \{pairs $\langle h_i, in_i \rangle$, where $in_i$ is the injection associated to the power set, and $h_i$ is in $\mathcal{H}(\mathcal{L} \times \mathcal{H}(N))$. Each $h_i$ is thus either in $\mathcal{L} \times \mathcal{H}(N)$, or a pair $\langle q_i, n_i \rangle$, where the “bound” name $n_i$ is the fresh name received by a bound input transition. Each $q_i$ in turn is an element of the product $\mathcal{L} \times \mathcal{H}(N)$. Thus, each $q_i$ is either in the form $\langle l, q, in_1, in_2 \rangle$, or in the form $\langle l, \langle q, n \rangle, in_1, in_2 \rangle$, where $l \in \mathcal{L}$, $q \in Q_N$, $n \in \parallel q \parallel_N$. The former is a transition which is not a bound output, while the latter is a bound output transition whose fresh name is $n$. This representation of transitions is the same that was employed in [17]. In particular the construction of quadruples (§3.2.1 therein), that represent single transitions of the $\pi$-calculus, is quite similar to elements of $\mathcal{L} \times \mathcal{H}(-)$. 

\textbf{Remark 8.1} In [20], a finitary endofunctor for the early semantics of the $\pi$-calculus is given, built from $\delta$, products, finite power set and a name exponential. Translating this functor to named sets would result in a functor not requiring normalisation. Further study is required on the subject; in particular, it is unclear how partition refinement works in that case, as redundant transitions must be eliminated to compute the minimal model.

We also remark that redundant transitions have an impact on efficiency of analysis algorithms, e.g. when an exhaustive enumeration of all the paths of the automaton is needed. The number of paths can grow as $n^k$ where $n$ is the number of names in the support of each state, and $k$ is the (possibly huge) number of states of the system. Consider e.g. the $\pi$-calculus agent (parametrised over $k \in \omega$) $P^k$ recursively defined as $P^1 = \emptyset$, and $P^{i+1} = a(x).P^i$ for $i > 0$. $P^k$ reaches all the $k$ states $P^i$ for $i \in \{1, \ldots, k\}$. At each step, an input transition is performed. The outgoing transitions from each state are $n + 1$ whenever $n$ names are considered as part of the support, and the corresponding redundant transitions are not eliminated. Therefore, the number of paths in the automaton is $(n + 1)^k$. Notice that the only name that is actually necessary in the support is $a$, which is free in $P^k$, but e.g. the bisimilar agent $P^k \parallel (\nu z)z.w.\emptyset$ requires to add redundant transitions for $w$ to all states.

8.2 Petri Nets and Causal Bisimulation

In [31], causal automata were introduced to represent the history-preserving semantics of place-transition Petri nets of [4]. Here names play the role of
events that are generated each time a transition is fired, and can be subsequently referenced to denote a causal dependency. Names in each state are local, and bisimilarity is a ternary relation featuring pairs of states, and name mappings between them. In the case of Petri nets, in contrast to the π-calculus, it is natural to use local names, since these denote locally generated events. In the definitions below, we omit the initial states of causal automata. While this is not a big change for the theory of causal automata, adding the possibility to observe initial states in coalgebraic bisimulation is not correct, as it distinguishes them from non-initial, but possibly bisimilar states.

**Definition 8.2 (Causal automaton)** Given a set Act of labels, a causal automaton is a tuple \(\langle Q, w, \rightarrow \rangle\) where \(Q\) is a set of states, \(w : Q \rightarrow \mathcal{P}_{\text{fin}}(\omega)\) associates to each state a finite set of names, and \(\rightarrow\) is a finite set of transitions. Each transition has the form \(q \xrightarrow{\alpha} q'\), where \(q, q' \in Q\) are the source and target states, \(\alpha \in \text{Act}\) is the label, \(D \subseteq w(q)\) are the dependencies of the transition, and \(\sigma : w(q') \mapsto w(q) + 1\) is an injective renaming for the transition. The additional name \(* \in 1\) is a freshly generated name denoting the current transition in the target state.

**Definition 8.3 (Causal bisimulation)** A set of triples \(R\) is a causal bisimulation between \(\langle Q_N, w_N, \rightarrow \rangle\) and \(\langle Q_M, w_M, \rightarrow' \rangle\) iff for all \(\langle q_N, d, q_M \rangle \in R\), the following holds: \(q_N \in Q_N, q_M \in Q_M\) and \(d\) is a partial bijection between \(w_N(q_N)\) and \(w_M(q_M)\); whenever \(q_N \xrightarrow{a,D} q_N'\), then \(d\) is defined on \(D\), there exists some transition \(q_M \xrightarrow{a,d(D)} q_M'\) and some \(d'\) such that \(\langle q_N', d', q_M' \rangle \in R\), and \(d'(n) = m\) implies \(\sigma_N(n) = \sigma_M(m) = *\) or \(d(\sigma_N(n)) = \sigma_M(m)\); whenever \(q_M \xrightarrow{a,D} q_M'\), then \(d^{-1}\) is defined on \(D\), there exists some transition \(q_N \xrightarrow{a,d^{-1}(D)} q_N'\) and some \(d'\) such that \(\langle q_N', d', q_M' \rangle \in R\), and \(d'(n) = m\) implies \(\sigma_N(n) = \sigma_M(m) = *\) or \(d(\sigma_N(n)) = \sigma_M(m)\).

Causal automata can be viewed as coalgebras in \(\text{NSet}\) for the functor \(C(N) = \mathcal{P}_{\text{fin}}(\text{Act} \times \mathcal{P}_{\text{fin}}(\omega)) \times \text{H}(N)\). The named set \(\text{Act}\) has no names. \(\mathcal{P}_{\text{fin}}(\omega)\) is seen as a named set with \(S(p) = \{id_p\}\), denoting the causal dependencies of each transition.

In this section we do not use symmetries; therefore in the following we use injective functions rather than singletons of injective functions to represent arrows of \(\text{Symset}\). Also, since \(\text{Act}\) has no names, the categorical product of this named set with any other is actually a “degenerate” set of triples, in the sense that there are only empty name mappings, hence we can avoid to denote these in the product \(\text{Act} \times \mathcal{P}_{\text{fin}}(\omega)\) and just use pairs of elements.

**Definition 8.4 (From causal automata to C-coalgebras)** Given a causal automaton \(A = \langle Q, w, \rightarrow \rangle\), we define the corresponding coalgebra \(\text{tr}_A : N_A \rightarrow C(N_A)\) as follows. The named set of states is \(N_A = \langle Q, S \rangle\), where \(S(q) = \)
\{id_{w(q)}\}$, that is, the symmetry of each state is trivial, and the support is given by the function $w$.

To define $tr_A = \langle h, \Sigma \rangle$, for $q \in Q$, consider the set of its outgoing transitions (indexed by a finite set $I$) $\{\langle a_i, q'_i, D_i, \sigma_i \rangle \mid q \xrightarrow{a_i D_i}_{\sigma_i} q'_i \land i \in I\}$. For each $i$, let $\rho_i$ denote the inclusion of $D_i$ into $w(q)$, and $\sigma' = \langle \sigma_i \rangle_{w(q'_i) \setminus (\sigma_i^{-1}(\cdot))}$ the injection of $w(q'_i)$, excluding the freshly generated name $\sigma_i^{-1}(\cdot)$, into $w(q)$ (notice the common target of these two injections). Let $in_1^i$ and $in_2^i$ denote $MCP(\langle \rho_i, \sigma' \rangle)$ in $\text{Symset}$, and let $t_i$ denote the unique mediating arrow. Finally, let $\langle \langle in_i, j \rangle \rangle_{i \in I}$ denote $MCP(\langle t_i \rangle_{i \in I})$ in $\text{Symset}$, and $j$ the corresponding unique arrow. Then $h(q) = \{\langle \langle a_i, D_i, q'_i, \sigma_i^{-1}(\cdot), in_1^i, in_2^i \rangle \mid i \in I \} \text{ and } \Sigma(q) = j$.

For each transition of the causal automaton, we identify the pair $\langle q'_i, \sigma_i^{-1}(\cdot) \rangle$, that is, the element of $H(N)$ that abstracts name $\sigma_i^{-1}(\cdot)$ in the destination state $q'_i$. The names of the label and causal dependencies $\langle a_i, D_i \rangle$ and of the state (having an abstracted name) $\langle q_i, \sigma_i^{-1}(\cdot) \rangle$ are related by the cospan $\langle \rho_i, \sigma' \rangle$. Its canonical representative is the multi-coproduct of the two arrows, making the quadruple $\langle \langle a_i, D_i, q'_i, \sigma_i^{-1}(\cdot), in_1^i, in_2^i \rangle \rangle_{i \in I}$ an element of $\text{Act} \times \mathcal{P}_\text{fin}(\omega) \times H(N)$. The unique arrow $t_i$ maps the names of this element to the names of the source state $q$. All the arrows $t_i$ (one for each transition) have a common codomain (the names of $q$), thus, the whole (finite) set of transitions is an element of the finite power set because of the multi-coproduct on all the $t_i$, with associated tuple of injections $\langle \langle in_i \rangle \rangle$. The obtained unique mapping $j$ maps names of the whole set of transitions to names of the source state $q$, thus it is used as $\Sigma(q)$.

In the following, we denote with $\iota$ the (obviously defined) isomorphism that given a partial injective mapping $d : n \to m$ returns a cospan $\langle \langle in_1, in_2 \rangle \rangle \in \text{MCP}(n, m)$ (in $I$, thus also in $\text{Symset}$) so that $d(i) = j \iff \iota_{in_1} (i) = \iota_{in_2} (j)$. A bisimulation between two $C$-coalgebras, that is, a subset of the product of the two named sets of states, can be easily obtained from a causal bisimulation using $\iota$.

**Theorem 8.5** Given two causal automata $N$ and $M$, a set of triples $R$ is a bisimulation between them if and only if $\{\langle q_N, q_M, in_1, in_2 \rangle \mid \langle q_N, d, q_M \rangle \in R \land \langle in_1, in_2 \rangle = \iota(d)\}$ is a coalgebraic bisimulation between $tr_N$ and $tr_M$.

9 Conclusions

We have introduced a theory of accessible endofunctors in the category of named sets, acting as a compositional specification language for algebras and coalgebras with name allocation and garbage collection of unused names.
Among the future work, the minimisation procedure of [17] should be generalised to the new framework, and a model checking algorithm should be developed. For the latter, we first need to define an appropriate logic characterizing the model, possibly taking advantage of Stone duality as in [6]. The minimisation algorithm of [17] computes the greatest symmetry of a system up to bisimulation. Since permutation groups can be described in logarithmic space (w.r.t. their size) using their generators, and many useful operations on permutation groups can work directly on this compact representation (see [27]), these algorithms should be exploited to obtain an efficient model checking algorithm. A comparison with the results of [38] may lead to a uniform theory of symmetry in model checking.

Besides working on names, it is possible to develop the presheaf semantics of a given calculus using richer index categories, such as the category of finite cardinals and all functions F used in [28], the categories of distinctions [24], or a suitable category to model name fusions (a starting point for this can be found in [5]). It would be interesting to see if and when the equivalence between the Schanuel topos and named sets resists to the generalisation, and if the functors presented here (in particular, the finitely representable power set) have a more general form.

Finally, we mention the possibility to use named sets to develop more syntax-oriented applications, e.g. parser generators for languages with binders. It would be interesting to see what classes of languages can be specified and recognised using nominal techniques.

References


A Proofs

Proof A.1 (prop. 3.3) For the symmetry, we have \( \pi \in \mathcal{G}_{\delta(A)}(a) \iff \pi^{-1}(a) = a \iff \pi^{-1} \in \mathcal{G}_A(a) \). Now we show that \( S_a = \{i - 1 \mid i \in \text{supp}_A(a) \setminus \{0\}\} \) supports \( a \) in \( \delta(A) \): \( \pi \in \text{fix}(S_a) \implies \forall i \in \text{supp}_A(a) \setminus \{0\}, \pi(i - 1) = i - 1 \implies \forall i \in \text{supp}_A(a), \pi^{-1}(i) = i \implies \pi^{-1} \in \text{fix}(\text{supp}_A(a)) \implies \pi^{-1} \in \mathcal{G}_A(a) \implies \) (by the result on the symmetry) \( \pi \in \mathcal{G}_{\delta(A)}(a) \). Finally, suppose that \( X \subseteq S_a \) supports \( a \) in \( \delta_A \). Let \( Y = \{0\} \cup \{i + 1 \mid i \in X\} \), so that by definition of \( S_a \) we have \( \text{supp}_A(a) \notin Y \). We show that \( Y \) supports \( a \) in \( A \), thus getting to a contradiction: \( \bar{\pi} \in \text{fix}(Y) \implies \) (since \( \bar{\pi}(0) = 0 \) \exists \pi, \bar{\pi} = \pi^{-1} \land \pi^{-1} \in \text{fix}(Y) \implies \) (by hypothesis) \( \pi \in \text{fix}(X) \implies \pi \in \mathcal{G}_{\delta(A)}(a) \implies \) (by the result on \( \mathcal{G}_{\delta(A)} \)) \( \pi^{-1} \in \mathcal{G}_A(a) \implies \bar{\pi} \in \mathcal{G}_A(a) \).

Proof A.2 (prop. 3.8) Since old and hidden elements are disjoint, each element \( a \) of \( \delta(A) \) is represented by an element of \( A^a \). We show that the orbits of all elements in \( H^a \) are disjoint. First, observe that \( i \in \text{supp}_A(a) \implies \text{old}_A(a) \notin \text{orb}_{\delta(A)}(\text{hid}^a(a)) \) because of the different cardinality of the supports: by Thm. 3.3 \( \text{supp}_{\delta(A)}(\text{hid}^a(a)) = \text{supp}_A(a) \setminus \{i\} = \text{supp}_{\delta(A)}(a) \setminus \{i\} \). Hence the orbit of the old element is disjoint from the orbit of any hidden element. Now we show that \( \text{orb}_{\delta(A)}(h_i(a)) = \text{orb}_{\delta(A)}(h_j(a)) \iff \exists \pi \in \mathcal{G}_A(a), \pi(i) = j \). For each \( i \in \text{supp}_A(a) \) we have \( \text{orb}(h_i(a)) = \{\pi_A^{-1}(h_i(a)) \mid \pi \in \text{Autf}\} = \{\pi_A(h_i(a)) \mid \pi \in \text{Autf} \land \pi(0) = 0\} = \{\pi \circ \pi' \} \land (0 \land \pi'(0) = 0 \land \pi'(i) = 0 \land \forall j \in \text{supp}(a) \setminus \{i\}, \pi'(j) = j + 1\} = \{\pi_A(a) \mid \pi(i) = 0\} \). Finally, suppose that \( \text{orb}_{\delta(A)}(h_i(a)) = \text{orb}_{\delta(A)}(h_j(a)) \). For each \( \pi \) such that \( \pi(j) = 0 \), there exists \( \pi' \) such that \( \pi'(i) = 0 \) and \( \pi_A(a) = \pi'_A(a) \). Then \( a = (\pi^{-1} \circ \pi')_A(a) \) iff \( \pi^{-1} \circ \pi' \in \mathcal{G}_A(a) \). Observing that \( \pi^{-1} \circ \pi'(j) = i \) we conclude our proof.

Proof A.3 (prop. 3.14) For the “only if” part, let \( A = \langle A, \pi_A \rangle \) be a permutation algebra, and \( f : A \to T(A) \). We show that \( f \) respects the transition specification \( \Delta_n \). Outgoing transitions from an element \( a \in A \) are either in the form \( \langle 0, (l, a) \rangle \), where \( a \in A \) and \( l \in L' \), or in the form \( \langle 1, \{\text{bt}(x), a\} \rangle \), with a element of \( \delta(A) \). In the first case, the action of a permutation on a transi-
tion respects rule 1: \( \langle 0, (l,a) \rangle \in f(a) \iff \rho_{\mathcal{L} \times \mathcal{A}}((0, (l,a))) \in f(\rho_{\mathcal{A}}(a)) \iff (0, \rho(l), \rho(a)) \in f(\rho_{\mathcal{A}}(a)) \). In the second case the action of a permutation respects rule 2: we have \( \langle 1, (\text{bout}(x), a) \rangle \in f(a) \iff \rho_{\mathcal{L} \times \delta(\mathcal{A})}(\langle 1, (\text{bout}(x), a) \rangle) \in f(\rho_{\mathcal{A}}(a)) \iff (1, \rho(\text{bout}(x)), \rho^{+1}(a)) \in f(\rho_{\mathcal{A}}(a)) \).

For the “if” part, observe that if a transition function is in \( \text{FSAlg}^\pi \), then it is a transition system with a finitely supported set of transitions. This is because finitely supported permutation algebras are a full subcategory of permutation algebras, hence the transition function has to preserve finiteness of the support. Obeying to either meta-rule 1 or 2, and being a labelled transition system, brings in the coproduct of products. The transition function, being a morphism, has to preserve the permutation action, and meta-rules specify exactly the permutation action in the destination state of a transition. Rule 1 specifies that the permutation action is unchanged, hence we obtain the identity functor. Rule 2 specifies that the permutation action must act, in the target, as \( \rho^{+1} \) for each permutation which acts as \( \rho \) in the source. This is the definition of \( \delta \).

**Proof A.4** (prop. 5.3) We omit the proof that \( \mathcal{H} \) is a functor. First we note the following lemma, which links the support and symmetry of elements of \( \mathcal{G}(N) \) to the properties of elements of the named set \( N \).

**Lemma A.5** The support of an element of the algebra \( \mathcal{G}(N) \) is obtained as \( \text{supp}_{\mathcal{G}(N)}(\langle q, \rho \circ \mathcal{S}_N(q) \rangle) = \rho(\text{dom}(\mathcal{S}_N(q))) \). The symmetry of an element of \( \mathcal{G}(N) \) is obtained as \( \text{sym}_{\mathcal{G}(N)}(\langle q, \rho \circ \mathcal{S}_N(q) \rangle) = \rho \circ \mathcal{S}_N(q) \circ \rho^{-1} \).

Next, we define a natural isomorphism \( i \) such that, for \( K : N \to M \), \( \delta(\mathcal{G}(K)) \circ \iota_N = \iota_M \circ \mathcal{G}(H(K)) \). We have \( \iota_N(\langle q, \rho \circ \mathcal{S}_H(q) \rangle) = \rho_{\mathcal{G}(N)}(\text{old}_G(\langle q, \mathcal{S}_N(q) \rangle)) \) and \( \iota_N(\langle (q, i), \rho \circ \mathcal{S}_H(q) \rangle) = \rho_{\mathcal{G}(N)}(\text{hid}_G(\langle q, \mathcal{S}_N(q) \rangle)) \). By Thm. 2.9, it suffices to show that \( i \) is an isomorphism between canonical representatives of orbits that preserves and reflects the symmetry of elements. The orbits of \( \langle q, \mathcal{S}_H(q) \rangle \) and \( \langle (q, i), \mathcal{S}_H(q) \rangle \) are syntactically distinguished. We take these as canonical representatives.

By Thm. 3.8, the orbits of \( \text{old}_G(\langle q, \mathcal{S}_N(q) \rangle) \) and \( \text{hid}_G(\langle q, \mathcal{S}_N(q) \rangle) \) are distinguished in turn. All the elements of \( \mathcal{G}(H(N)) \) are covered by the above two cases by definition of \( H \), and all elements of \( \delta(\mathcal{G}(N)) \) are in the image of \( \iota_N \) as old and hidden elements are disjoint. Thus, \( i \) is an isomorphism of orbits. We show that the symmetry is reflected and preserved. By Lemma A.3, the symmetry of \( \langle q, \mathcal{S}_H(N)(q) \rangle \), restricted to its support, is \( \mathcal{S}_H(N)(q) = \mathcal{S}_N(q) \), and the symmetry of \( \langle (q, i), \mathcal{S}_H(N)(q, i) \rangle \), again restricted to its support, is \( \text{gf} \mathcal{S}_N(q, i) \) by Definitions 5.1, 5.2, and Lemma A.5 again. By Thm. 3.3 and Lemma A.3, the symmetry of \( \text{old}_G(N)(\langle q, \mathcal{S}_N(q) \rangle) \) restricted to its support is \( \mathcal{S}_N(q) \) and the symmetry of \( \text{hid}_G(\langle q, \mathcal{S}_N(q) \rangle) \) is \( \text{gf} \mathcal{S}_N(q, i) \).

We now show that the isomorphism is natural, i.e. it commutes with arrows (it suffices to show this for canonical representatives). For elements without hidden names, the proof is trivial. For an element \( \langle (q, i), \mathcal{S}_H(N)(q, i) \rangle \), we have two cases: either \( \exists \sigma \in \Sigma_K(q) \sigma(j) = i \) or not. In the first case, let \( \Sigma_K(q) = \rho \circ \mathcal{S}_M(h_K(q)) \). We have \( \iota_M(\mathcal{G}(H(K))(\langle (q, i), \mathcal{S}_H(N)(q, i) \rangle)) = \ldots \).
\[ \iota_M(\langle h_K(q), j \rangle, \rho \circ g \cdot \text{fix } (S_M(q), j)_{l_{M,S_M}}) = \rho_{G(M)}(\text{hid}^n_{G(M)}(\langle q, S_M(q) \rangle)). \]

Moreover, \( \delta(G(K))(\nu_N(\langle \langle q, i \rangle, S_{H}(N)(\langle q, i \rangle) \rangle)) = \delta(G(K))(\text{hid}^n_{G(N)}(\langle q, S_N(q) \rangle)) = \rho_{G(M)}(\text{hid}^n_{G(M)}(\langle q, S_M(q) \rangle))). \) The other case is similar.

**Proof A.6** (prop. 6.3) For each named set \( C \), and arrows \( F : C \to N \), \( G : C \to M \), we define the unique morphism from \( C \) to \( N \times M \) as \( \langle h, \Sigma \rangle \), where \( h(q \in Q_C) = \langle h_F(q), h_G(q), \langle in_1, in_2 \rangle \rangle \), and \( \Sigma(q) = u \).

Here \( \langle in_1, in_2 \rangle \in \text{MCP}(\langle S_N(h_F(q)), S_M(h_G(q)) \rangle) \), with a unique morphism commuting with the cocone \( (\Sigma_F(q), \Sigma_G(q)) \in \text{Symset} \). The set-theoretical part of the construction is unique since it corresponds to the set theoretical product. The name mappings are also unique, from uniqueness of the multi-coproduct.

**Proof A.7** (prop. 6.9) Let \( \langle p, \rho \circ S(p) \rangle \in G(\mathcal{P}_{\text{cat}}(N)) \), with \( p = \{q_i, in_i \mid i \in I\} \in \mathcal{P}_{\text{cat}}(N) \). The inverse of \( \iota \) is \( \iota^{-1}(\langle p, \rho \circ S(p) \rangle) = \{q_i, \rho \circ in_i \mid i \in I\} \), where \( \rho \) is such that \( \rho_{|\mathcal{P}} = \rho \). A choice must be fixed for \( \rho \in \rho \circ S(p) \) and \( \rho \) that agrees with \( \rho \) on \( \|p\| \); this choice does not matter as we now see. We have \( \iota(\iota^{-1}(\langle p, \rho \circ S(p) \rangle)) = \iota(\{q_i, \rho \circ in_i \mid i \in I\}) = \langle p, \rho \circ S(p) \rangle = \langle p, \rho \circ S(p) \rangle \).

Naturality comes from commutativity of the multi-coproducts in Def. 6.8.

**Proof A.8** (prop. 6.14) It is easy to see that \( \iota \) is an isomorphism due to the normalisation condition on the representation functor. We now have to prove that \( \iota \) is a natural transformation, that is, it commutes with each arrow \( f : A \to B \). First observe that for each \( i \in I \) we have \( f(\text{close}_A(a_i, n_i)) = \text{close}_B(f(a_i), n_i) \).

The simplest case is when \( \forall i, j \in I \).close\( _B(f(a_i), n_i) \cap \text{close}_A(f(a_j), n_j) = 0 \). In this case, we trivially have \( (\iota \circ \mathcal{P}_{\text{fin}})(p)) = (\mathcal{P}_{\text{fin}}(\delta(f)) \circ \iota)(p) \).

If the hypothesis does not hold, then \( \exists i, j \in I \).close\( _B(f(a_i), n_i) \cap \text{close}_A(f(a_j), n_j) \neq 0 \). The proof then is done by inspection of nine cases, depending on the conditions \( n_i \in \text{supp}_A(a_i), n_i \in \text{supp}_B(f(a_i)), n_j \in \text{supp}_A(a_j), n_j \in \text{supp}_B(f(a_j)) \). The cases should actually be more, but the fact that \( n \notin \text{supp}_A(a) \implies n \notin \text{supp}_B(f(a)) \) rules out some possibilities.

The corner case, where normalisation plays a role, is when \( n \notin \text{supp}_B(f(a_i)) \) and \( n_j \in \text{supp}_B(a_j) \). Then, we have \( \text{close}_B(f(a_i), n_i) = \{f(a_i)\} \), and \( f(a_i) \in \text{close}_B(f(a_i), n_j) \), thus there exists \( n \) such that \( \sigma^{n,n_i}_B(f(a_i)) = a_i \). On the other hand, we have \( \delta(f)(a_i) = \text{old}_B(f(a_i)) \), and \( \delta(f)(\text{hid}^n(a_j)) = \text{hid}^n(f(a_j)) \), but norm will remove the “spurious” element \( f(a_i) \), since \( \text{hid}^n(f(a_i)) = \text{hid}^n(f(a_j)) \), thus commutativity of the natural transformation is obtained.

**Proof A.9** (prop. 7.2) The product and coproduct are accessible in \( \text{Set} \) and are constructed “pointwise”, that is, using the set-theoretical definition over the carriers, in \( \text{FSAlg}^n \). Therefore they preserve the same limits in \( \text{Set} \) and \( \text{FSAlg}^n \). The countable power set is accessible in \( \text{Set} \), and constructed pointwise in \( \text{Alg}^n \). The finitely supported version of \( \text{FSAlg}^n \) is a subfunctor of it, having in turn \( \mathcal{P}_{\text{fin}} \) as a subfunctor. By [13], Corollary 6.31, all the subfunctors of an accessible functor are accessible. A direct proof that \( \mathcal{P}_{\text{fin}} \) is actually finitary may also be given, observing that the functor is determined by its action
on the algebras having a finite number of orbits, but it is not required here.

**Proof A.10 (prop. 8.5)** The key observation is that the correspondence $\iota$ is a bijection. Then notice that using $\delta$ constrains the fresh names to be identified in coalgebraic bisimulation, thus mimicking causal bisimulation. The rest of the proof just comes from commutativity of the multi-coproduct.