Università di Pisa
Dipartimento di Informatica

Technical Report: TR-09-24

Minimal support and families for the semantics of calculi with structured resources

Vincenzo Ciancia , Alexander Kurz , Ugo Montanari

December 9, 2009

Address: Largo B. Pontecorvo 3, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

Minimal support and families for the semantics of calculi with structured resources

Vincenzo Ciancia *, Alexander Kurz **, Ugo Montanari ***

Abstract. Calculi that feature resource-allocating constructs (e.g. the pi-calculus or the fusion calculus) require special kinds of models. The best-known ones are presheaves and nominal sets. But named sets have the advantage of being finite in a wide range of cases where the other two are infinite. The three models are equivalent. Finiteness of named sets is strictly related to the notion of finite support in nominal sets and the corresponding presheaves. We generalise previous equivalence results by introducing a notion of minimal support in presheaf categories indexed over small categories of monos. We show that nominal sets are generalisd by families, that is, free coproduct completions, indexed by symmetries. Functors and categories of coalgebras may be defined over *families*. We show that the final coalgebra has the greatest possible symmetry up-to bisimilarity, which can be computed by iteration along the terminal sequence, thanks to finiteness of the representation.

1 Introduction

Full abstraction and nominal calculi. One of the greatest concerns in programming language semantics is to find *fully abstract* models, where all the semantically equivalent programs are identified. A difficult question in this area is how to identify such models for the so-called *interactive systems*, where the focus is not the final result of the computation, but rather on the interactions with the environment along the possibly non-terminating *behaviour* of a system. For languages such as the CCS [31] or the π-calculus [32], the operational semantics is expressed in terms of *labelled transition systems* (LTS), and the fully abstract model is the quotient of all the possbile systems with respect to *bisimilarity*.

Among the reasons to look for such models we mention not only the possibility of proving properties of programs, but also the applications to *finite state methods* such as *equivalence checking*, *model checking* and *minimisation* of systems. Algorithms to perform these tasks are well-known for ordinary (finite state) labelled transition systems; in particular, minimisation and equivalence checking can be done by partition refinement [36], while a wide range of algorithms for model checking exists, depending on the logic, and on a plethora of optimisation techniques.

* Universidad Complutense de Madrid
** University of Leicester
*** Università di Pisa

Calculi with resource allocation mechanisms (the so called *nominal calculi*) often have a notion of bisimulation that does not coincide with the standard one in LTS. Thus, standard definitions and algorithms can not be reused. This is solved by resorting to presheaf categories, that is, categories of functors from a small category `C` to `Set` [20,8,7,21,30,29], or to *finitely supported permutation algebras* [33], that is, the *nominal sets* of Gabbay and Pitts [22]. Interestingly, the latters were introduced to solve the problems arising in abstract syntax with binding, and in particular from the axioms of α-conversion. Both syntax and semantics are turned into standard notions (resp. initial algebras and final coalgebras) using these kinds of models. The operational semantics of a calculus in presheaf models typically has infinite states even for very simple processes, making it difficult to *compute* the abstract semantics, or to *implement* model checking algorithms. Presheaves handle names, and in general resources, as having a *global* meaning across all possible processes: the link between the names of different processes is established *a priori*. Thus, name generation must be implemented in such a way that each generated name has its own, unique meaning.

Named sets. In the parallel research line of *named sets* [35,37], these difficulties were overcome using *local* names; in this case, establishing a binding between names of elements is necessary whenever two elements are related. This machinery allows one to reuse previously generated names that have been discarded. In [37], a number of formalisms including *place-transition Petri nets* equipped with *history preserving bisimulation*, the *CCS with localities* and the π-calculus have been mapped into named sets in a fully abstract way. The most important finding here is that modelling the *symmetry* of agents is necessary to have a unique abstract model of the π-calculus. In [17,39,18], a minimsation algorithm for the π-calculus has been implemented based on the model of *history-dependent automata*, that is, coalgebras in the category of *named sets*. Remarkably, the computed minimal system, which is the fully abstract model, that is, the *final coalgebra*, contains an explicit representation, in terms of its *generators*, of the greatest permutation group over names of an agent that preserves bisimulation. Finally, in [41], a number of *ad-hoc* constructions on named sets used for the π-calculus are turned into categorical notions such as products, coproducts, the power set and name abstraction (also detailed in [10]), thus allowing one to reuse the same machinery to represent the semantics of other calculi with names. The importance of modelling symmetries has been recently recognised both in the theory of programming language semantics [42] and in practical applications such as model checking [15], making a generic symmetry reduction algorithm appealing for the efficient verification of interactive systems. We remark that, due to well known properties of group theory (in particular Lagrange's theorem, see e.g. [14], §3.3), the symmetry has an efficient representation in terms of generators: there exists a representation of each finite group which is logarithmic with respect to the size of the group. Moreover, many operations on groups can be computed on the compressed representation [28].

The categorical equivalence between three models of nominal computation, that is, nominal sets, named sets and the *pullback-preserving* full subcategory[1] of $\mathtt{Set}^{\mathtt{I}}$, called the *Schanuel topos*, has been established in [24,19].

Our contributions. A great advantage of presheaf categories is the additional flexibility that can be obtained by varying the index category $\mathtt{C}$, giving rise to models that do not have only *pure names*, but rather more complex structures (see e.g. [25], or [3]). However, the already mentioned problems related to the intrinsically infinite nature of presheaf models limit the practical applicability of obtained result. As an example, consider a program that allocates a resource y, makes this observable by sending it over a public channel x, and immediately restarts using y as the public channel and generating a new resource, looping indefinitely. This is exemplified by the π-calculus agent $P(x) = (\nu y)\bar{x}y.P(y)$. The semantics of this agent in nominal sets and presheaves is not a loop, but rather an infinite sequence of allocation actions, each one going into a different agent, because the fresh names in agents must be all different. Therefore it is not possible e.g. to run a model checker on such a semantics. In the named sets representation, this agent is a loop. Each transition has an associated injective relabelling, that allows the semantics to reuse the same fresh name in all the steps (the interested reader may refer to [10] where this example is developed in detail). This is a prototypical example of how the representation that we advocate introduces a notion of *garbage collection* that allows the semantics to reuse existing resources assigning to them a new meaning.

Perhaps the most important topic in [22] is the notion of *finite support*, which generalises the notion of *free variables* in terms. The support is in turn the key ingredient to define named sets. In this work, we define a more general notion of support. Exploiting this definition, we show that the equivalence result of [24,19] can be extended to presheaves indexed by small categories, respecting three conditions: the index category has wide pullbacks, and the presheaves preserve them; the index category is made up of monos; all the arrows of the index category from an object to itself are isomorphisms.

In §3 we show a representation of a presheaf category that is based on *families* (that is, free coproduct completions) of categories of symmetry groups. Presheaves are presented as sets of *elements* that have an attached symmetry on their available *local interfaces*. The representation we propose removes all the redundant information that is present in a presheaf because of inclusions and isomorphisms of stages. In particular, the representation can be finite even if the elements of the presheaf form an infinite set. Also, the advantage of named sets when modelling name generation is immediately generalised, because elements that are obtained from isomorphic objects of the index category are identified.

The three conditions that we mentioned may seem to limit the applicability of the framework. Of these, the first two seem more important, giving to the presheaf category the necessary structure to remove the redundant information.

[1] Here $\mathtt{I}$ is the category of finite subsets of the natural numbers and injections between them.

A natural question here is to understand when non-monic index categories are really necessary in the semantics of programming languages, and when it is possible to find equivalent representations that fall under our hypothesis (e.g. in [3] a category having just monic arrows is used to model *explicit fusions*).

Moreover, the results in §§4-5, that we now introduce, do *not* depend on these conditions (in particular, on the arrows of the index category being monos), but rather they are properties of families in general, and are the main reason why we consider families appealing for computer science. An eventual generalisation of the conditions we require would still have these properties.

Presheaves and families have a very different nature. We refer to this as *locality of interfaces*. In §4 we give a mathematical explanation of this property, which is reflected in the product construction. The product is just computed *point-wise* in presheaves, while it involves a mapping of the local interfaces of each involved element into a greater one, in the case of families. This corresponds to two radically different, though equivalent, views on how systems with interfaces may be related, namely by either assuming a *naming authority* giving a global meaning to each available resource, or by relying on locally scoped *links* that connect the different systems. The former is the view of the π-calculus and related literature, while the latter may be more useful in modelling decentralised systems such as *sensor networks* or *peer to peer systems*. In §5, we show how to compute the *behavioural symmetry* of an element of a coalgebra, that is, the greatest group of isomorphisms that leave an element bisimilar to itself. This is done by iteration along the terminal sequence, exploiting the "finitistic" nature of the representation. A detailed analysis of the algorithm and the possibility to take advantage of generators of symmetry groups for complexity improvements are left as future work. Finally, in §6 we provide the link to the existing theory, explaining how the equivalence of [24,19] is an instance of our framework.

Related work. The work presented here directly extends [24,19]. The most closely related work is [2], whose Thm. 3.3 is very similar to our Thm. 2. However, they require at least that the index category is *accessible* and has an initial object, and the proof is carried out only in the case when the category is locally presentable. On the other hand, our paper is focused on generalising the finite support condition of Gabbay and Pitts to other index categories, and exploits this to give a representation inspired by the model of named sets. The proof we present does not have the requirements of [2], and in particular does not rely on existence of an initial object, therefore it works uniformly also when the index category is a disconnected category, or as a special case, a discrete category, that is, a multi-sorted set. This clarifies that one result is not a consequence of the other (the theorem of [2] can not be in turn a consequence of our result, since it also deals with index categories that are not small). The intended application of the theory we develop is to represent the semantics of process calculi in a finite way, and to provide a generic minimization and symmetry-reduction procedure. Indeed, this aspect is not studied at all in [2], whose aim is to unify various mathematical notions that generalise the *analytic functors* of Joyal.

2 Background

Here we introduce the basic notions related to the *family* construction, which is a representation of the *free coproduct completion* of a category.

Remark 1. (notational conventions). For $\mathtt{C}$ a category, we denote with $|\mathtt{C}|$ its objects, with $\mathtt{C}(n,m)$ the set of arrows from n to m. We extend some categorical notations to *sets* of arrows. Let $F \subseteq \mathtt{C}(n,m)$ be a set; we define $dom(F) = n$ and $cod(F) = m$. When F and G are two such sets, with $dom(F) = cod(G)$, $f : cod(G) \to m'$, and $g : m'' \to dom(F)$, we define $f \circ G = \{f \circ g \mid g \in G\}$, $F \circ g = \{f \circ g \mid f \in F\}$, and $F \circ G = \{f \circ g \mid f \in F, g \in G\}$. As a notation for the elements of the coproduct $\coprod_{x \in S} P_x$ in $\mathtt{Set}$, we use the set of pairs $\{\langle x, p\rangle \mid x \in S, p \in P_x\}$. The copairing of a tuple of arrows $f_{i \in I}$ is denoted with $\coprod_{i \in I} f_i$. We often omit the parenthesis in function and functor application, e.g. we write $\mathtt{F}fx$ to denote the action of the functor $\mathtt{F} : \mathtt{C} \to \mathtt{Set}$ on the arrow f, applied to the element x. With *pullbacks* we actually refer to *wide*, but small, pullbacks.

A direct description of the free coproduct completion of a category $\mathtt{C}$ is obtained by the *family* construction, defined as follows.

Definition 1. *Given a small category* $\mathtt{C}$, objects *of the category* $\mathtt{Fam}(\mathtt{C})$ *are* families *of objects of* $\mathtt{C}$*, that is, coproducts* $\coprod_{i \in I}\{n_i\}$ *of singletons in* $\mathtt{Set}$*, where* I *is a set, and, for each* $i \in I$, $n_i \in |\mathtt{C}|$. *An* arrow *from* $\coprod_{i \in I}\{n_i\}$ *to* $\coprod_{j \in J}\{m_j\}$ *is a tuple* $\langle f, \coprod_{i \in I}\{\mathcal{H}_i^f\}\rangle$*, where* $f : I \to J$ *and, for each* $i \in I$, $\mathcal{H}_i^f : n_i \to m_{f(i)}$.

A family is a set I, where each element $i \in I$ has an associated $\mathtt{C}$-object n_i. The set I may represent, for example, the set of states of a system. The object n_i represents the *interface* of the state i. For example, n_i can be a set of names, a network topology, or any other possible feature associated to the states of a process calculus. Each arrow is a function f between two sets I and J, and for each $i \in I$ there is a map $\mathcal{H}_i^f$ from the interface of i to that of $f(i)$. This reflects the idea that interfaces are *local* to each element, therefore to properly define a function between such elements, one also has to specify how the interfaces of destination and source elements are related. When we use families to represent presheaves, as we shall see, these maps go in the other direction, that is, from the destination to the source. Looking at the above definition, this does not make a big difference, as one can just consider the category $\mathtt{Fam}(\mathtt{C}^{op})$ to get these "backwards" arrows, as we shall do in the following. A real-world example of local interfaces which can help the intuition is the injective relabelling of memory locations that may happen after an invocation of the garbage collector in a garbage-collected language. System states in this case have an associated memory layout (its "interface" in our terminology), that may change at each step of the execution. The relabelling is the "backward" arrow that we mention, mapping the memory layout of the destination into that of the source, thus tracking the history of variables and their memory locations along the computation.

There is a canonical embedding of $\mathtt{C}$ into $\mathtt{Fam}(\mathtt{C})$, which is also full.

Definition 2. *The embedding* $\mathtt{H} : \mathtt{C} \to \mathtt{Fam}(\mathtt{C})$ *acts on objects as* $\mathtt{H}n = \coprod_{i \in 1}\{n\}$, *and on arrows* $f \in \mathtt{C}(n, m)$ *as* $\mathtt{H}h = \langle id_1, \coprod_{i \in 1}\{h\}\rangle$.

For completeness, we recall that coproducts in $\mathtt{Fam}(\mathtt{C})$ are *freely* generated, and each object of $\mathtt{Fam}(\mathtt{C})$ is a coproduct of objects in the image of $\mathtt{H}$.

Definition 3. *The* coproduct *in* $\mathtt{Fam}(\mathtt{C})$ *of two objects* $\coprod_{i \in I}\{n_i\}$ *and* $\coprod_{j \in J}\{m_j\}$ *is defined as* $\coprod_{k \in I+J}\{o_k\}$, *where* $o_k = n_i$ *if* $k = \langle I, i\rangle$, *and* $o_k = m_j$ *if* $k = \langle J, j\rangle$.

3 Representing pullback-preserving presheaves

Presheaves are functors from a small *index* category to $\mathtt{Set}$. We start from the point of view that objects of the index category are *interfaces*, or *specifications* [40], that *systems* expose, by which one can manipulate the semantics of a program or compose subsystems into a larger system. The monos of the index category allow an interface to be embedded into a larger one, for composition purposes. If the index category has pullbacks, and the functors preserve those, then we have a similar situation to the *finite support* condition in the work by Gabbay and Pitts on nominal syntax [23]: each system has a unique "proper interface", from which all the larger ones can be recovered using the operations. Such systems may be represented in two alternative ways; as presheaves, that is, multi-sorted collections of elements, and operations over them, or as enriched sets, with a *local* interface attached to elements. The latter notion is formalised by the means of *families*, that we advocate as an alternative, "more computational" abstract model for the operational semantics of calculi.

The fundamental idea of representing presheaves by families of symmetries comes from Staton [38], where it appears as a proof technique to show that named sets and the Schanuel topos are equivalent. The technical results that we present in this section are a direct generalisation of that work, even though the purposes are different, since we aim to explain the computational properties of the families model, which is done in the rest of the paper.

From now on, we let $\mathtt{C}$ be a small category, and $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ the wide-pullback-preserving full subcategory of $\mathtt{Set}^{\mathtt{C}}$. Our theory can be instantiated under the following conditions, that we assume in the rest of the paper:

- all the arrows of $\mathtt{C}$ are monic;
- $\mathtt{C}$ has (small, wide) pullbacks;
- for every object n of $\mathtt{C}$, each $f \in \mathtt{C}(n, n)$ is an isomorphism.

Notice that we do *not* require strong properties on $\mathtt{C}$ e.g. completeness or co-completeness. Some examples may clarify the applicability of the framework.

Discrete categories: the one-object and one-arrow category $\mathtt{1}$ can be used as an index, resulting in a degenerate instantiation of the framework that actuall just contains sets and functions. This is correct, as $\mathtt{Set}^{\mathtt{1}}$ is $\mathtt{Set}$. More generally, *discrete* categories can be used, in this case the representation that we will define is just the set of elements of each presheaf, that is, pairs $\langle n, x\rangle$

where n is the index where x lives. This is a very natural representation of *multi-sorted* sets. These two examples show that the definition works also in these degenerate cases, giving the expected representation.

Finite sets and injections: in this case, the obtained equivalence is that between the Schanuel topos and named sets of [19,24]. The associated categories have been used in a wide range of applications as we already emphasized.

Finite graphs and injections: this category can be used to model calculi whose network structure is made explicit in the semantics (as opposed to the π-calculus, where the network structure is left implicit in the knowledge of channels by agents) and whose semantics is closed with respect to adding links to the network. An example calculus with an explicit network topology, that will be object of investigation, is the *network coordination policies* calculus [9], whose states are pairs consisting of the network topology, represented as a graph, and a policy, which is a program.

Fusions: the category $\mathtt{F}$ of finite sets and all functions has been used in [29] to model the fusion calculus. This category is not included in our framework since it is not made up of monos, but more prominently the category in question does not have wide pullbacks. Therefore it seems unlikely to obtain a representation in the spirit of this work, even if the restriction to monos was not present. It is possible to use a different approach: fusions may also be described by the means of monos, if the *objects* of the index category are rich enough. This is witnessed by [3], where fusions are modelled using a category of equivalence relations that only has monic arrows. However, in that case the index category does not have all pullbacks. We plan to investigate the usage of the category of relations over a finite domain, and monic, monotone maps between them. This category has pullbacks, falls into the conditions of our framework, but it has a rich structure of objects that may be used for fusions (see also [25,30]).

3.1 The category $\mathtt{Sym}(\mathtt{C})$

Definition 4. *We define the (small) category* $\mathtt{Sym}(\mathtt{C})$ *of* symmetries over $\mathtt{C}$:

$$|\mathtt{Sym}(\mathtt{C})| = \bigcup_{n \in |C|} \{\Phi \subseteq \mathtt{C}(n,n) \mid \Phi \textit{ is a group w.r.t. composition}\}$$

$$\mathtt{Sym}(\mathtt{C})(\Phi_1, \Phi_2) = \{h \circ \Phi_1 \mid h \in \mathtt{C}(dom(\Phi_1), dom(\Phi_2)) \wedge \Phi_2 \circ h \subseteq h \circ \Phi_1\}$$

The identity of each object is defined as $id_\Phi = id_{dom(\Phi)} \circ \Phi = \Phi$*; the composition law is given by* $(h_2 \circ \Phi_2) \circ (h_1 \circ \Phi_1) = h_2 \circ h_1 \circ \Phi_1$.

Arrows of the category are sets of arrows from $\mathtt{C}$, obtained by composition of a group of isomorphisms with a single arrow. It is very important to observe that the composition symbol on the left hand side of the last equation is the composition in $\mathtt{Sym}(\mathtt{C})$ which is being defined, and not composition of set of arrows, while the composition on the right is composition of sets of arrows, as from Remark 1. However the following lemma ensures that the two possible interpretations coincide. This is a consequence of the condition $\Phi_2 \circ h \subseteq h \circ \Phi_1$.

Lemma 1. *Consider two* $\mathtt{Sym(C)}$ *arrows* $h_2 \circ \Phi_2 : \Phi_2 \to \Phi_3$ *and* $h_1 \circ \Phi_1 : \Phi_1 \to \Phi_2$. *It holds that* $(h_2 \circ h_1) \circ \Phi_1 = \{h_2 \circ \varphi_2 \circ h_1 \circ \varphi_1 \mid \varphi_2 \in \Phi_2 \wedge \varphi_1 \in \Phi_1\}$.

Finally we note that $\mathtt{C}$ has a full embedding into $\mathtt{Sym(C)}$.

Definition 5. *The embedding* $\mathtt{J} : \mathtt{C} \to \mathtt{Sym(C)}$ *is defined on objects as* $\mathtt{J}(n) = \{id_n\}$ *and on arrows as* $\mathtt{J}(f) = \{f\}$.

3.2 The equivalence between $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ and $\mathtt{Fam}(\mathtt{Sym(C)}^{op})$

We employ the following well-known proposition (see [6], Lemma 42), used in [38], to prove the equivalence between named sets and the Schanuel topos.

Proposition 1. *Let* $\mathtt{D}$ *be a locally small category having small coproducts, and* $\mathtt{C}$ *a small category. A functor* $\mathtt{F} : \mathtt{C} \to \mathtt{D}$ *can be extended to an equivalence from* $\mathtt{Fam(C)}$ *to* $\mathtt{D}$ *if it satisfies the following conditions:* $\mathtt{F}$ *is an embedding (it is injective on objects and morphisms); objects in the image of* $\mathtt{F}$ *are indecomposable (for each object* n *of* $|\mathtt{C}|$*, the homset functor* $\mathtt{D}(\mathtt{F}n, -)$ *preserves coproducts); every object of* $\mathtt{D}$ *is a coproduct of objects in the image of* $\mathtt{F}$.

We now exhibit a functor $\mathtt{F} : \mathtt{Sym(C)}^{op} \to \mathtt{Set}^{\mathtt{C}}_{\Diamond}$.

Definition 6. *The functor* $\mathtt{F} : \mathtt{Sym(C)}^{op} \to \mathtt{Set}^{\mathtt{C}}_{\Diamond}$ *acts on each object* Φ *returning a presheaf in* $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ *as* $\mathtt{F}\Phi n = \{h \circ \Phi \mid h \in \mathtt{C}(dom(\Phi), n)\}$*, and* $\mathtt{F}\Phi f(h \circ \Phi) = f \circ h \circ \Phi$. $\mathtt{F}$ *acts on each arrow* $h \circ \Phi_1 : \Phi_2 \to \Phi_1$ *of* $\mathtt{Sym(C)}^{op}$ *returning a natural transformation, defined at each index* n *as* $(\mathtt{F}(h \circ \Phi_1))_n(h' \circ \Phi_2) = h' \circ h \circ \Phi_1$.

The intuition here is that the functor $\mathtt{F}$ reconstructs a presheaf "almost freely" starting from a symmetry Φ, that is, the action of each operation h is freely generated, but the set of possible arrows h is quotiented by composition with Φ, meaning that two arrows h and h' such that $h' = h \circ \rho$ for $\rho \in \Phi$ are identified.

We show that $\mathtt{F}$ is a functor from $\mathtt{Sym(C)}^{op}$ to $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$.

Theorem 1. $\mathtt{F}$ *is a functor and for each* Φ*,* $\mathtt{F}\Phi$ *preserves wide pullbacks.*

As $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ has coproducts, $\mathtt{F}$ can be extended to a covariant functor from $\mathtt{Fam}(\mathtt{Sym(C)}^{op})$ to $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$, making one direction of the categorical equivalence.

Definition 7. *The functor* $\mathtt{Presh} : \mathtt{Fam}(\mathtt{Sym(C)}^{op}) \to \mathtt{Set}^{\mathtt{C}}_{\Diamond}$ *is defined as*

$$\mathtt{Presh} \coprod_{i \in I} \{\Phi_i\} = \coprod_{i \in I} \mathtt{F}\Phi_i \qquad \mathtt{Presh}\langle f, \coprod_{i \in I} \{\mathcal{H}_i^f\}\rangle = \coprod_{i \in I} (\iota_{f(i)} \circ \mathtt{F}\mathcal{H}_i^f)$$

where $\langle f, \coprod_{i \in I} \{\mathcal{H}_i^f\}\rangle : \coprod_{i \in I} \{\Phi_i\} \to \coprod_{j \in J} \{\Phi'_j\}$*, and* $\iota_{f(i)}$ *denotes the* $f(i)^{th}$ *injection of the coproduct* $\coprod_{j \in J} \mathtt{F}\Phi'_j$.

Next, we show that $\mathtt{F}$ respects the first and second conditions of Prop. 1. First, recall that if $\mathtt{C}$ is small, the functor category $\mathtt{Set}^{\mathtt{C}}$ is locally small and has coproducts (defined pointwise), hence Prop. 1 is applicable.

Proposition 2. $\mathtt{F}$ *is an* embedding*, i.e. injective on objects and morphisms.*

Proposition 3. *For each object* $\Phi : \mathtt{Sym(C)}$*,* $\mathtt{F}\Phi$ *is indecomposable, that is, the homset functor* $\mathtt{Set}^{\mathtt{C}}_{\Diamond}(\mathtt{F}\Phi, -)$ *preserves coproducts.*

3.3 The symmetric decomposition of a presheaf

In this section we conclude the proof that $\mathtt{F}$ respects the conditions of Prop. 1, by providing a characterisation of functors in $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$. First, we recall the notion of *element* of a presheaf. Hereafter, we let $\mathtt{G}$ denote an arbitrary functor in $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$.

Definition 8. *The set of* elements *of* $\mathtt{G}$ *is defined as* $El(\mathtt{G}) = \coprod_{n \in |\mathtt{C}|} \mathtt{G}n$.

For readability, but without loss of generality, in the following we assume that all the $\mathtt{G}n$ are disjoint, so that we are able to denote with just x the element $\langle n, x\rangle \in El(\mathtt{G})$. When necessary, we denote the stage n of x as $st(x)$.

Roughly, we aim to represent presheaves by quotienting all the elements that are "reachable" from some common element by the action of arrows. To make this formal, we introduce the notion of *orbit*.

Definition 9. *Given* $x \in El(\mathtt{G})$*, its* orbit $\mathcal{O}_x$ *is the set of elements* $y \in El(\mathtt{G})$ *such that there exist a span* $st(x) \xleftarrow{f_x} s \xrightarrow{f_y} st(y)$ *and an element* $z \in \mathtt{G}s$*, with* $\mathtt{G}f_x z = x$ *and* $\mathtt{G}f_y z = y$.

In other words, an orbit is a connected component in the *category of elements*. In the following, for $x \in El(\mathtt{G})$, we let D^x be the diagram in $\mathtt{C}$ consisting of the morphisms $\{d : n \to st(x) \mid \exists y \in \mathtt{G}(n).\mathtt{G}dy = x\}$, for n ranging over $|\mathtt{C}|$. Notice that, for each d, y is uniquely determined: $\mathtt{G}d$ is injective because $\mathtt{G}$ is pullback-preserving, hence mono-preserving.

The following lemma forms the grounds of our representation. It is perhaps the most important property of orbits, due to pullback preservation of $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$.

Lemma 2. *Let* x *and* y *belong to the same orbit. Let* n *be the pullback object of* D^x *and* m *be the pullback object of* D^y*. There exists an isomorphism between* n *amd* m *making* n *a pullback of* D^y.

We now define the *support* of an element x, which is, roughly speaking, the smallest index where an element having the same properties of x can be found.

Definition 10. *Let* $x^{\mathcal{O}}$ *denote a choice of an element in* $\mathcal{O}_x$*. We define the* support *of* x*, denoted with* $\mathcal{S}_x$*, as the pullback object of* $D^{(x^{\mathcal{O}})}$*, and the* normalising arrow $\mathcal{N}_x : \mathcal{S}_x \to st(x)$ *as the diagonal of the pullback diagram of* D^x*, where we choose* $\mathcal{S}_x$ *as the pullback object by Lemma 2.*

With *diagonal* here we mean the composition of any arrow in D^x with the corresponding arrow making the pullback commute.

We are going to see that an object of $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ is determined (up-to isomorphism) just by a set of representatives $\widehat{x}$ of elements, called *proper elements*, and by the set of isomorphisms over the stage of each $\widehat{x}$ whose action leaves $\widehat{x}$ unchanged. Preservation of pullbacks plays a fundamental role here, allowing us to prove the following lemma and to define the representative of an element.

Lemma 3. *There exists a unique element* $\widehat{x} \in \mathtt{G}\mathcal{S}_x$ *such that* $\mathtt{G}\mathcal{N}_x\widehat{x} = x$.

Definition 11. *Let $x \in El(\mathtt{G})$. We denote with $\widehat{x}$ the* representative *of x, that is, the element of $\mathtt{G}\mathcal{S}_x$ such that $\mathtt{G}\mathcal{N}_x(\widehat{x}) = x$. The set of* proper elements *of $\mathtt{G}$ is defined as $Pel(\mathtt{G}) = \{\widehat{x} \mid x \in El(\mathtt{G})\}$.*

In this construction, $\mathcal{N}_x$ plays the role of a canonical arrow whose action recovers x from its representative $\widehat{x}$. The *symmetry* associates to each proper element an object of $\mathtt{Sym}(\mathtt{C})$.

Definition 12. *The* symmetry *of $\widehat{x} \in Pel(\mathtt{G})$ is the group of isomorphisms $\mathcal{G}_{\widehat{x}} = \{\rho : \mathcal{S}_x \to \mathcal{S}_x \mid \mathtt{G}\rho\widehat{x} = \widehat{x}\}$.*

Now we can define a functor from $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ to $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ which, together with the functor $\mathtt{Presh}$ of Def. 7, completes the categorical equivalence.

Definition 13. *The* symmetric decomposition $\mathtt{SymDec} : \mathtt{Set}^{\mathtt{C}}_{\Diamond} \to \mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ *is defined on each presheaf $\mathtt{G}$ and natural transformation $f : \mathtt{G}_1 \to \mathtt{G}_2$ as*

$$\mathtt{SymDec}(\mathtt{G}) = \coprod_{\widehat{x} \in Pel(\mathtt{G})} \{\mathcal{G}_{\widehat{x}}\} \qquad \mathtt{SymDec}(f) = \langle \lambda\widehat{x}.\widehat{f_{\mathcal{S}_x}(\widehat{x})}, \coprod_{\widehat{x} \in Pel(\mathtt{G}_1)} \{\mathcal{N}_{f(\widehat{x})} \circ \mathcal{G}_{\widehat{f_{\mathcal{S}_x}(\widehat{x})}}\}\rangle$$

The action of the functor on objects just records the proper elements of $\mathtt{G}$, and their symmetry. The action on arrows is an arrow of $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$, thus a function between the two index sets, and a family of arrows in $\mathtt{Sym}(\mathtt{C})^{op}$. The former returns, for each representative $\widehat{x}$, the representative of $f_{\mathcal{S}_x}(\widehat{x})$. The mappings associated to the arrow are the normalising arrows of every obtained element, composed with the corresponding symmetry. Using it, one can reconstruct $f_{\mathcal{S}_x}(\widehat{x})$ from its representative. A bit more intuition may be obtained by considering the support and symmetry of an element as a *local* interface of that element. The arrow $\mathcal{N}_{f(\widehat{x})} \circ \mathcal{G}_{\widehat{f_{\mathcal{S}_x}(\widehat{x})}}$ embeds the interface of $\widehat{f_{\mathcal{S}_x}(\widehat{x})}$ into the interface of $f_{\mathcal{S}_x}(\widehat{x})$, which is the same of $\widehat{x}$ because f is defined pointwise. The normalising arrow is the so-called *history of names along morphisms*[2] used in the literature on named functions, and in coalgebras it plays a similar role to the injective relabelling of memory locations done by garbage collectors in the implementation of programming languages.

Lemma 4. *We have $\widehat{\mathtt{G}h\widehat{x}} = \widehat{x}$, and $\mathcal{N}_{\mathtt{G}h\widehat{x}} \in h \circ \mathcal{G}_{\widehat{x}}$.*

Theorem 2. *Every presheaf $\mathtt{G}$ in $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ is isomorphic to $\mathtt{Presh}(\mathtt{SymDec}(\mathtt{G}))$.*

The main result of this section is the sum of Prop. 1, 2, 3 and Thm. 2.

Theorem 3. *The category $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ is equivalent to $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$.*

From now on, given an arbitrary object $P = \coprod_{i \in I}\{\Phi_i\}$ of $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$, we call I the set of *elements*, or *underlying set*, of P, and we say that Φ_i is the *symmetry* of the element $i \in I$. It is straightforward to check the following.

[2] In our case, we should call it the history of *interfaces* along morphisms.

Proposition 4. *The Yoneda embedding* $\mathtt{y} : \mathtt{C}^{op} \to \mathtt{Set}^{\mathtt{C}}$ *is equal to* $\mathtt{Presh} \circ \mathtt{H} \circ \mathtt{J}$ *where* $\mathtt{H}$ *and* $\mathtt{J}$ *come from Def. 2 and Def. 5.*

Remark 2. A great advantage of the proposed representation of presheaves using families is to reduce the size (the number of elements) of the represented presheaf, even getting a finite set out of an infinite one, while preserving the categorical properties. For example, the "inclusion" presheaf $\mathtt{G}n = n, \mathtt{G}f = f$ in $\mathtt{Set}^{\mathtt{I}}$, that is, the object of names in $\mathtt{Set}^{\mathtt{I}}$, is represented by a family having a single element in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{I})^{op})$, namely $\coprod_{i \in 1}\{id_1\}$. The intuitive meaning of this assertion is that each natural number is not distinguishable from any other, and has a single "name" (and trivial symmetry) as its interface. This "finitistic" representation is the main reason why named sets and history-dependent automata have been considered appealing for the static analysis of nominal calculi (model checking [26], and bisimulation checking [18]).

4 Locality of interfaces: the product construction

In [41], one of the authors extended the equivalence of [24,19] to the categories of coalgebras of equivalent endofunctors, in order to give a categorical characterisation of the various constructions that had been used in the past for named sets (including minimisation of the π -calculus). Here we generalise the result on representing the product in the category of named sets exploiting multi-coproducts in a suitable category (the category $\mathtt{Symset}$ that here is recalled in §6).

Multi-(co)products are a specialisation of the notion of multi-(co)limit, studied in detail by Diers [13]. It is well known (see e.g. [11], remark 5) that $\mathtt{Fam}(\mathtt{C})$ has products whenever $\mathtt{C}$ has multi-products, and dually, $\mathtt{Fam}(\mathtt{C}^{op})$ has products if $\mathtt{C}$ has multi-coproducts. Here we provide a concrete characterization of the functor, that emphasizes the difference between *global* and *local* interfaces. The results presented here do not rely on arrows of $\mathtt{C}$ being mono.

Definition 14. *Given a diagram D consisting of a tuple of objects $\langle n_1, \ldots, n_k \rangle$, the* multi-coproduct *of D is a set $mcp(D)$ of cocones over D such that for all cocones $L' = \langle f_1 : n_1 \to m', \ldots, f_k : n_k \to m' \rangle$ over D there exists a unique cocone $L = \langle \iota_1 : n_1 \to m, \ldots, \iota_k : n_k \to m \rangle \in mcp(D)$, and a unique arrow $u_{L'} : m \to m'$ making the diagram $L \cup L' \cup u_{L'}$ commute. The unique cocone L will be denoted, with a bit of overloading, with $mcp(L')$.*

In words, the multi-coproduct of two objects P and Q is a set of *canonical* cospans between them, in the sense that they are quotiented by isomorphisms of cospans, and they are minimal.

We note that $\mathtt{Sym}(\mathtt{C})$ has multi-coproducts.

Theorem 4. *If $\mathtt{C}$ has wide pullbacks, then $\mathtt{Sym}(\mathtt{C})$ has multi-coproducts.*

In the following definitions, we assume that $\mathtt{C}$ has multi-coproducts, that $P = \coprod_{i \in I}\{n_i\}$, $Q = \coprod_{i \in J}\{m_j\}$, $R = \coprod_{k \in K}\{o_k\}$ are three arbitrary objects of $\mathtt{Fam}(\mathtt{C}^{op})$, and we denote with S the set $\{\langle i, j, \langle \iota_1, \iota_2 \rangle \rangle \mid i \in I \wedge j \in J \wedge \langle \iota_1, \iota_2 \rangle \in mcp(\langle n_i, m_j \rangle)\}$.

Definition 15. *The* product *of P and Q in $\mathtt{Fam}(\mathtt{C}^{op})$ is defined as the object $P \times Q = \coprod_{\langle i,j,\langle \iota_1,\iota_2\rangle\rangle \in S}\{cod(\iota_1)\}$.*

Elements of the product $P \times Q$ are triples, formed by an element of P, an element of Q, and a (canonical) cospan relating their symmetry.

Definition 16. *Let π'_1 and π'_2 denote the first two projections of the ternary product S. The* projections *$\pi_1 : P \times Q \to P$ and $\pi_2 : P \times Q \to Q$ are defined as $\pi_1 = \langle \pi'_1, \coprod_{\langle i,j,\langle \iota_1,\iota_2\rangle\rangle \in S}\{\iota_1\}\rangle$, $\pi_2 = \langle \pi'_2, \coprod_{\langle i,j,\langle \iota_1,\iota_2\rangle\rangle \in S}\{\iota_2\}\rangle$.*

Definition 17. *The* pairing *of $\langle f, \coprod_{k\in K}\{\mathcal{H}^f_k\}\rangle : R \to P$ and $\langle g, \coprod_{k\in K}\{\mathcal{H}^g_k\}\rangle : R \to Q$ is the arrow $\langle h, \coprod_{k\in K}\{\mathcal{H}^h_k\}\rangle$, where $h(k) = \langle f(k), g(k), mcp(\langle \mathcal{H}^f_k, \mathcal{H}^g_k\rangle)\rangle$, and $\mathcal{H}^h_k = u_{\langle \mathcal{H}^f_k, \mathcal{H}^g_k\rangle}$.*

Theorem 5. *The product, projections and pairing given above identify up to isomorphism the binary product in $\mathtt{Fam}(\mathtt{C}^{op})$.*

In the above definition, $mcp(\langle \mathcal{H}^f_k, \mathcal{H}^g_k\rangle)$ and $u_{\langle \mathcal{H}^f_k, \mathcal{H}^g_k\rangle}$ come from Def. 14. We keep on with the intuition that the index category $\mathtt{C}$ in $\mathtt{Set}^{\mathtt{C}}$ should be perceived as a set of possible *types*, or *interfaces* of elements of the presheaf. In this light, the definition of the product above gives a notion of *locality* of interfaces in families, as opposed to a notion of *global* interfaces in presheaf categories.

In $\mathtt{Set}^{\mathtt{C}}$ the product is defined pointwise, and two elements may be related by just pairing them if they are in an appropriate (common) context. That is, any two interfaces have a natural choice of an embedding into a common, greater interface, thus their relative meaning is established once and for all. In the case of names (that is, where the index category is $\mathtt{I}$), this is the vision adopted by the π-calculus, where the names of all the non-restricted channels of an agent have a global, unique meaning across all participating parallel components of a system, as if there was a *naming authority* assigning a meaning to any name.

In $\mathtt{Fam}(\mathtt{C}^{op})$, whenever we put two elements in a relation, we have to explicitly establish a link between their interfaces by exhibiting them as subobjects of a common object, acting as the interface of the obtained tuple. In the case of names, this corresponds to having to "pull wires" among all parallel components of a system to make explicit how they can interact. This may be the most natural choice whenever one wants to model systems that do not have a naming authority, such as *peer-to-peer* systems.

As an example, bisimilarity in $\mathtt{Fam}(\mathtt{C}^{op})$ is made up of triples, because it is a subobject of the product: in order to compare two systems, we need to establish a correspondence between their local interfaces.

5 Symmetry reduction by final semantics

The presheaf approach to operational semantics consists in defining a presheaf P of *terms*, that is, the initial algebra of some endofunctor over a presheaf category, possibly quotiented with structural axioms, and a coalgebra from P to

$\mathtt{T}P$ for some endofunctor $\mathtt{T}$, providing the semantics of the calculus. The unique morphism into the final coalgebra of $\mathtt{T}$ then gives the coinductive definition of the abstract semantics. Here we link the symmetry of elements in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ with *behavioural equivalence*, defined as the pullback object of a coalgebra morphism. We note that coalgebraic bisimilarity and behavioural equivalence coincide if the behavioural functor $\mathtt{T}$ preserves weak pullbacks (see [27] or [1] for details). Given a coalgebra in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$, and an element i, having symmetry Φ with $dom(\Phi) = n$, we explain how computing the image of i along the unique morphism into the final coalgebra corresponds to identify the subobject of n that is *active* in the semantics of i, and the greatest possible symmetry over this object that preserves behavioural equivalence.

The interest of this result is in providing a clean framework (namely, the equivalence between presheaves and families) for symmetry reduction of the semantics of programming languages. Symmetry reduction is an actively researched topic in computer science that consists in finding compressed representations of systems that have a symmetry (see [12] and subsequent works, or the more recent [16]). This is typically done exploiting equations on the syntax of calculi, or by adding symmetry information "by hand" to models. The approach that we propose is radically different in the fact that it allows one to *compute* the behavioural symmetry, that is, the best symmetry up-to bisimulation. This is certainly wanted in all the cases where bisimulation is the equivalence relation of choice (e.g. static analysis in service oriented computing and model checking of Hennessy-Milner-like logics). Model checking can be performed efficiently in the presence of symmetry [15]. The material presented here does not depend on the condition that arrows are mono, hence it is stable under a possible generalisation.

5.1 Symmetry reduction

The following proposition is easy to prove (a proof is available in [41]). We assume in the following such a pair of equivalent endofunctors $\mathtt{T}'$ and $\mathtt{T}$.

Proposition 5. *Each endofunctor* $\mathtt{T}'$ *over* $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ *that has a final coalgebra has an equivalent endofunctor over* $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ *admitting a final coalgebra, obtained (up to isomorphism) as* $\mathtt{T} = \mathtt{SymDec} \circ \mathtt{T}' \circ \mathtt{Presh}$.

Remark 3. Even if for the scope of this work it suffices to define $\mathtt{T} = \mathtt{SymDec} \circ \mathtt{T}' \circ \mathtt{Presh}$, it may be necessary to have a *compositional* definition of $\mathtt{T}$ so that the elements of $\mathtt{T}(P)$ are derived from those of P. In the case of the product, for example, the definition of §4 is isomorphic to the one obtained from Prop. 5, but not the same. This topic has been studied in detail in [41].

We now observe that each natural transformation in $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ induces a symmetry on elements of its source, explicitly represented in the corresponding arrow of $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$. Consider a presheaf $\mathtt{G} = \coprod_{i \in I} \mathtt{F}\Phi_i$, a natural transformation $f : \mathtt{G} \to \mathtt{G}'$, and the corresponding arrow $\langle g, \coprod_{i \in I} \{\mathcal{H}_i^g\} \rangle : \coprod_{i \in I} \{\Phi_i\} \to \coprod_{j \in J} \{\Phi'_j\}$.

Definition 18. *Let R^f_n denote the relation coming from the kernel pair of the component f_n of f at n. Let $x \in \mathtt{G}n$. We call the set $\mathcal{G}^h_x = \{\rho : n \to n \mid \mathtt{G}\rho x R^f_n x\}$ the symmetry on x induced by f.*

Proposition 6. *For each $i \in I$, $n \in |\mathtt{C}|$, $h \circ \Phi_i \in \mathtt{F}\Phi_i n$, and $\rho : n \to n$, we have $(\mathtt{F}\Phi_i \rho(h \circ \Phi_i)) R^f_n (h \circ \Phi_i)$ if and only if $\rho \circ h \circ \mathcal{H}^g_i = h \circ \mathcal{H}^g_i$.*

Observe that $\rho \circ h \circ \mathcal{H}^g_i = h \circ \mathcal{H}^g_i$ implies that, for each h' in $h \circ \mathcal{H}^g_i$, there is an isomorphism $\rho' \in \Phi'_{g(i)}$ such that $\rho \circ h' = h' \circ \rho'$, that is, the symmetry induced by f is reflected in $\Phi'_{g(i)}$.

It is now obvious to observe that the symmetry induced by coalgebra morphisms respects bisimulation. When f is the unique morphism into the final coalgebra, the induced symmetry is the greatest possible such subset. We call it the *behavioural symmetry*. In this case, the arrows in $h \circ \mathcal{H}^g_i$ identify a subobject of n that intuitively is the *active* "sub-interface" of an element, i.e. operations that do not touch it may not affect the semantics. To make this more precise, observe that, for each $h' \in h \circ \mathcal{H}^g_i$, we either have $\rho \circ h' \neq h'$ or $\rho \circ h' = h'$. The first case is the one where the symmetry $\Phi'_{g(i)}$ actually plays a role. In the second case, as all the arrows in $h \circ \mathcal{H}^g_i$ are obtained by composition of h' with an arrow in $\Phi'_{g(i)}$, composition with ρ leaves *all* of them unchanged. Then ρ is acting in some sense *outside* of the subobject identified by $h \circ \mathcal{H}^g_i$. For example, when the index category is $\mathtt{I}$, the image of h is the set of *active names* of a system, that is, names that are observable in the final semantics.

5.2 Partition refinement as a generic symmetry reduction algorithm

Here and in the next section we explain how to compute bisimilarity on a subset of the terms of a calculus, if certain finiteness conditions hold.

Consider a calculus equipped with a semantics in $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$, that is a coalgebra $s : P \to \mathtt{F}P$ for a suitable endofunctor $\mathtt{F}$ and a presheaf P representing the terms of the calculus, as it is typical in the presheaf approach. As we have seen, there is a corresponding coalgebra $t : P' \to \mathtt{T}P'$ in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ of a suitable endofunctor $\mathtt{T}$ corresponding to $\mathtt{F}$.

The partition refinement in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ can be computed on an object $\coprod_{q \in Q}\{\mathcal{G}_q\}$ (intended to be a subobject of P' above) as follows. First, we give an abstract description of the general algorithm, then we explain in detail the single steps and discuss some finiteness conditions to compute them in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$.

Definition 19. *Coalgebraic partition refinement is defined as follows:*

Initialization: *Let $f = t$, let $h : \coprod_{q \in Q}\{\mathcal{G}_q\} \to 1$ be the unique morphism into the final object of $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$, and z the unique morphism from $\mathtt{T}1$ to 1.*

Coinductive step(f, h, z): *If z restricted to $Im(\mathtt{T}h \circ f)$ is an isomorphism in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ then return $\mathtt{T}h \circ f$. Otherwise let $f' = \mathtt{T}f \circ f$, $h' = \mathtt{T}h$, $z' = \mathtt{T}z$, and compute **Coinductive step(f', z', w')**.*

Correctness of the algorithm is well known by the theory of coalgebras (see e.g. [43]). An intuition can be given as follows. At the nth iteration of the algorithm, the kernel of $\mathtt{T}h \circ f : \coprod_{q \in Q}\{\mathcal{G}_q\} \to \mathtt{T}^n 1$ is a partition of Q, which quotients elements that have the same observations in n steps. At each step, this partition is refined, that is, possibly split, according to the observations made in the nth iteration of the system. When z is an isomorphism, a fixed point is reached, and it is guaranteed that in all successives steps, the partition will remain unchanged. Therefore, the elements of Q that are equalised by $\mathtt{T}h \circ f$ are bisimilar. The isomorphism z is a subobject of the final coalgebra that represents the behaviour of the elements of Q.

The pairs of bisimilar systems in Q are described by the kernel pair of the final value of the arrow $\mathtt{T}h \circ f$, and the behavioural symmetry of each element $q \in Q$ is reflected in the symmetry of its image along the same arrow. When $\mathtt{C}$ is the free category over one object and $\mathtt{T} = \mathcal{P}_{fin}(L \times -)$, then $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ is $\mathtt{Set}$, L is a set of labels, and the algorithm is the classical partition refinement for labelled transition systems. When $\mathtt{C}$ is $\mathtt{I}$, there is a suitable endofunctor such that the algorithm above is the partition refinement procedure for the π-calculus of [34,18] (see [41] for the details).

Computing the semantics

To be able to compute partition refinement, we first need to describe the final object in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C}^{op}))$. In a similar fashion to Thm. 5, the final object in $\mathtt{Fam}(\mathtt{C})$ is a family of *multi-initial* objects, that is, a set MI of $\mathtt{C}$-objects such that for each object c of $\mathtt{C}$ there is a unique element $i \in MI$ and a unique arrow $u : i \to c$. Similarly to Thm. 4, it is possible to show that if $\mathtt{C}$ has pullbacks, then $\mathtt{Sym}(\mathtt{C})^{op}$ has a set of multi-initial objects.

Proposition 7. *Given a set MI of multi-initial objects in $\mathtt{Sym}(\mathtt{C})$, the object $P = \coprod_{\Phi \in MI}\{\Phi\}$ is a final object in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$. The unique arrow from $\coprod_{j \in J}\{\Phi_j\}$ to P is $\langle \lambda j.i_{\Phi_j}, \coprod_{\Phi \in \Phi_j}\{u_{\Phi_j}\}\rangle$, where i_{Φ_j} and u_{Φ_j} denote respectively the unique element of MI and the unique arrow corresponding to Φ_j in MI.*

It holds that if a category has an initial object i, then the singleton $\{i\}$ is a family of multi-initial objects. For example, in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{I})^{op})$ the final object is a singleton equipped with the trivial symmetry $id_\emptyset$, and arrows from other objects map all their elements to the unique one.

Getting back to partition refinement, to compute h, z and f one needs that Q is finite and that from each object of q the corresponding element of the final object is computable. We obviously assume that f is computable in each step of the algorithm (otherwise the problem would not be solvable). Similarly to the set-theoretical (classical) case, partition refinement needs that the family of reachable elements from Q is finite. Indeed, this condition is not always satisfied in Turing-equivalent calculi. However, static constraints may be imposed (e.g. the *finite-control* π-calculus agents of [18]). Notice that the elements in the presheaf representation of Q need not be finite; finiteness in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ is

only finiteness of the number of *orbits* of a presheaf. The most important example where the $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ representation is finite, while the elements of the presheaf are not, is that of systems allocating new resources in a loop, discarding old resources so that the number of active resources is bound (see §5.3 and [10]).

One also needs that the image of f is finite on all the elements of Q, in order to be able to enumerate the elements on which z has to be an isomorphism. This requirement is certainly satisfied if $\mathtt{T}$ sends finite families into finite families. This happens in many interesting cases, including polynomial functors, name allocation, and certain non finite subfunctors of the power set. Remarkably, in [41] such a "finitistic" representation is given for the *early semantics* of the π-calculus, which is defined as an infinitary transition system, as input transitions are quantified over all possible names.

Under the above restrictions, one has to check if $z = \langle f_z, \coprod_{i \in Im(\mathtt{T}hof)} \{\mathcal{H}_i^{f_z}\}\rangle$ is an isomorphism. The criterion in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ is that f_z is an isomorphism in $\mathtt{Set}$ and each $\mathcal{H}_i^{f_z}$ is an isomorphism in $\mathtt{Sym}(\mathtt{C})$. To check the latter, it is necessary to determine the symmetry of elements of $\mathtt{T}^n 1$ for each n. Having an effective procedure to compute this symmetry depends on the chosen functor. In [41] it is shown how to do this for polynomials, name abstraction and subfunctors of the power set. We conjecture that these results generalise to other categories of finite structures, such as finite graphs.

5.3 Garbage collection

We consider the representation using families appealing because it may allow one to implement iteration along the terminal sequence on the operational semantics, in the presence of fresh resouce allocation. We emphasize that fresh resources are perhaps the most important reason to employ presheaves for the semantics of programming languages.

In presheaf models, whenever behavioural functors that may *allocate* new resources, such as the functor δ for name abstraction of [21], are used to build coalgebras, the operational semantics obtained by rules typically becomes infinite even in very simple cases. Again, this comes from the fact that interfaces have a *global* meaning in presheaves, whereas in the family representation the symmetry of each element is *local*. This is reflected in the definition of arrows: in presheaves, one does not need to provide information on how the interface of the destination is mapped in the interface of the source, while this is exactly the role of the family of arrows in $\mathtt{Sym}(\mathtt{C})$ (one for each element) that are the second component of an arrow of $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$. Thus, elements that have the same behaviour up-to an operation on their interface are not identified using presheaves. This is particularly problematic for recursive processes that allocate some resources while discarding older ones, keeping a finite quantity of resources allocated in each state (as explained in [10]). Using families, on the other hand, by virtue of the universal quantification over all arrows of Def. 11, all these equivalent elements are identified. It is the purpose of the family of maps associated to each arrow of the category to identify a subobject of the interface of each source state, which is preserved in the destination state, thus discarding unused resources.

The functor $\mathtt{SymDec}$ maps an operational semantics defined on presheaves to a canonical representation featuring a form of "garbage collection" along morphisms, allowing iteration along the terminal sequence to converge in these cases.

6 Named Sets and the Schanuel Topos

Here we show how theorem 2 generalises the equivalence between two models of name passing: *named sets* and the *Schanuel Topos* [24,19,38], that is, the pullback-preserving full subcategory of $\mathtt{Set}^{\mathtt{I}}$. For this, we adopt the "structured" and simplified notation of named sets developed in [41], that we recast in our framework. There, a category of permutation groups and injective relabellings quotiented by these groups, called $\mathtt{Symset}$, is used.

Definition 20. *The category* $\mathtt{Symset}$ *is the category* $\mathtt{Sym}(\mathtt{I})^{op}$*, where* $\mathtt{I}$ *is the category of finite subsets of the set of natural numbers* ω*, whose morphisms are total injective functions.*

Named sets are then defined as pairs $\langle Q, S\rangle$ where Q is a set of *elements* and S is a function from Q to $|\mathtt{Symset}|$. Named functions are pairs $\langle h, \Sigma\rangle : \langle Q_1, S_1\rangle \to \langle Q_2, S_2\rangle$ where $h : Q_1 \to Q_2$ and for each $q \in Q_1$, $S_2(q) \in \mathtt{Symset}(h(q), q)$. Indeed the definition can be paraphrased as follows.

Definition 21. *The category* $\mathtt{NSet}$ *is the category* $\mathtt{Fam}(\mathtt{Sym}(\mathtt{I})^{op})$.

Hence, the equivalence result of [24,19] is recovered as an instance of Thm. 3: $\mathtt{NSet}$ is equivalent to the Schanuel topos. In this light, we expect that the algorithmic benefits of named sets, explained in detail in [10], will extend to other cases, e.g. calculi indexed by graphs.

7 Concluding remarks

We have presented a framework to represent the semantics of programming languages that deal with *resources* or *interfaces* attached to system states: coalgebras over presheaf categories obeying to certain constraints, that give rise to a "finitistic" representation. This representation removes the redundant information coming from the notion of interfaces being *global* rather than *local*.

We can sketch some open questions. Removing the third condition should result in families of monoids instead of groups; understanding whether there are computational implications similar to symmetry reduction in the case of monoids is of interest. Resource allocation is an hot topic in the semantics of programming languages. Applications are of great interest in the area of *service-oriented computing*, where resource allocation in the presence of *network topologies* [9], or constraints [5] is an active field of research, and finite representations are of vital importance for the implementation of analysis algorithms. An efficient implementation of the generic symmetry reduction algorithm that we have presented should be studied. For that, one may take advantage of algorithms on

permutation groups exploiting the generators [28]. Finally, similar consideration apply to model checking. The study of a Stone-type duality for coalgebras over families in a similar fashion to [4], and a corresponding model checking algorithm exploiting the cases where the representation is finite, are one of our most important long-term goals.

It is expected that the categorical equivalence that we presented, combining the ease of specifying the semantics using presheaves with the implementative advantages of named sets, will enable the development of a general framework to specify (using presheaves) and analyse (using families) the semantics of calculi that have richer interfaces than pure names, thus advancing the research line of named sets and history dependent automata.

References

1. J. Adamek. Introduction to coalgebra. *Theory and Applications of Categories*, 14(8):157–199, 2005.
2. J. Adamek and J. Velebil. Analytic functors and weak pullbacks. *Theory and Applications of Categories*, 21(11):191–209, 2008.
3. F. Bonchi, M. Buscemi, V. Ciancia, and F. Gadducci. A Category of Explicit Fusions. *Lecture Notes in Computer Science - P. Degano and R. De Nicola and J. Meseguer, editors - Festschrift for Ugo Montanari*, 5065, 2008.
4. M. M. Bonsangue and A. Kurz. Pi-calculus in logical form. In *LICS*, pages 303–312. IEEE Computer Society, 2007.
5. M. G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In R. De Nicola, editor, *ESOP*, volume 4421 of *LNCS*, pages 18–32. Springer, 2007.
6. A. Carboni and E. Vitale. Regular and exact completions. *Journal of Pure and Applied Algebra*, 125(1-3):79 – 116, 1998.
7. G. L. Cattani and P. Sewell. Models for name-passing processes: Interleaving and causal. In *LICS*, pages 322–332, 2000.
8. G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the π-calculus. In *Category Theory and Computer Science*, pages 106–126, 1997.
9. V. Ciancia, G. L. Ferrari, R. Guanciale, and D. Strollo. Global Coordination Policies for Services. In *Workshop on Formal Aspects of Component Software (FACS) - ENTCS (upcoming)*, 2008.
10. V. Ciancia and U. Montanari. A name abstraction functor for named sets. *Electr. Notes Theor. Comput. Sci.*, 203(5):49–70, 2008.
11. C. Cirstea. Semantic constructions for the specification of objects. *Theor. Comput. Sci.*, 260(1-2):3–25, 2001.
12. E. M. Clarke, E. A. Emerson, S. Jha, and A. P. Sistla. Symmetry reductions in model checking. In A. J. Hu and M. Y. Vardi, editors, *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*, volume 1427 of *Lecture Notes in Computer Science*, pages 147–158. Springer, 1998.
13. Y. Diers. Familles universelles de morphismes. *Ann. Soc. Sci. Bruxelles*, 93:175–195, 1979.
14. J. D. Dixon and B. Mortimer. *Permutation Groups*, volume Permutation Groups of *Graduate Texts in Mathematics*. Springer, 2006.

15. E. A. Emerson and A. P. Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1/2):105–131, 1996.
16. E. A. Emerson and T. Wahl. Dynamic symmetry reduction. In N. Halbwachs and L. D. Zuck, editors, *TACAS 2005*, volume 3440 of *Lecture Notes in Computer Science*, pages 382–396. Springer, 2005.
17. G. L. Ferrari, U. Montanari, and M. Pistore. Minimizing transition systems for name passing calculi: A co-algebraic formulation. In *FoSSaCS*, pages 129–158, London, UK, 2002. Springer-Verlag.
18. G. L. Ferrari, U. Montanari, and E. Tuosto. Coalgebraic minimization of hd-automata for the pi-calculus using polymorphic types. *Theor. Comput. Sci.*, 331(2-3):325–365, 2005.
19. M. Fiore and S. Staton. Comparing operational models of name-passing process calculi. *Inf. Comput.*, 204(4):524–560, 2006.
20. M. P. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the pi-calculus (extended abstract). In *LICS*, pages 43–54, 1996.
21. M. P. Fiore and D. Turi. Semantics of name and value passing. In *LICS*, pages 93–104, 2001.
22. M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In G. Longo, editor, *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS'99)*, pages 214–224, Trento, Italy, 1999. IEEE Computer Society Press.
23. M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.
24. F. Gadducci, M. Miculan, and U. Montanari. About permutation algebras, (pre)sheaves and named sets. *Higher-Order and Symbolic Computation*, 19(2-3):283–304, 2006.
25. N. Ghani, K. Yemane, and B. Victor. Relationally staged computations in calculi of mobile processes. *Electr. Notes Theor. Comput. Sci.*, 106:105–120, 2004.
26. S. Gnesi and G. Ristori. A model checking algorithm for π-calculus agents. In *Proc. Second International Conference on Temporal Logic (ICTL '97)*. Kluwer Academic Publishers, 1997.
27. A. Kurz. *Logics for Coalgebras and Applications for Computer Science*. PhD thesis, Ludwig-Maximilians-Universitat Munchen, 2000.
28. E. M. Luks. Permutation Groups and Polynomial Time Computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.
29. M. Miculan. A categorical model of the fusion calculus. *Electr. Notes Theor. Comput. Sci.*, 218:275–293, 2008.
30. M. Miculan and K. Yemane. A unifying model of variables and names. In V. Sassone, editor, *FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2005.
31. R. Milner. A calculus of communicating systems. *Lecture Notes in Computer Science*, 92, 1980.
32. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part i. *Information and Computation*, 100(1):1–40, 1992.
33. U. Montanari and M. Pistore. pi-calculus, structured coalgebras, and minimal hd-automata. In M. Nielsen and B. Rovan, editors, *MFCS*, volume 1893 of *Lecture Notes in Computer Science*, pages 569–578. Springer, 2000.
34. U. Montanari and M. Pistore. Structured coalgebras and minimal hd-automata for the pi-calculus. *Theoretical Computer Science*, 340:539–576, 2005.

35. U. Montanari, M. Pistore, and D. Yankelevich. Efficient minimization up to location equivalence. In *ESOP*, pages 265–279, 1996.
36. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, December 1987.
37. M. Pistore. *History Dependent Automata.* PhD thesis, Università di Pisa, Dipartimento di Informatica, 1999. available at University of Pisa as PhD. Thesis TD-5/99.
38. S. Staton. Name-passing process calculi: operational models and structural operational semantics. Technical Report UCAM-CL-TR-688, University of Cambridge, Computer Laboratory, 2007.
39. E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming.* PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003. TD-8/03.
40. S. Vickers and G. Hill. Presheaves as configured specifications. *Formal Asp. Comput.*, 13(1):32–49, 2001.
41. Vincenzo Ciancia. *Accessible Functors and Final Coalgebras for Named Sets.* PhD thesis, University of Pisa, 2008.
42. G. Winskel. Symmetry and concurrency. In *CALCO*, pages 40–64, 2007.
43. J. Worrell. Terminal sequences for accessible endofunctors. *Electr. Notes Theor. Comput. Sci.*, 19, 1999.

A Appendix: Proofs

Proof (Lemma 1). First observe that $\Phi_2 \circ h \subseteq h \circ \Phi_1$ implies, as Φ_1 is a group, that $\Phi_2 \circ h \circ \Phi_1 \subseteq h \circ \Phi_1 \circ \Phi_1 = h \circ \Phi_1$. On the other hand, we have $h \circ \Phi_1 = id \circ h \circ \Phi_1 \subseteq \Phi_2 \circ h \circ \Phi_1$ (because Φ_2 contains the identity). Hence $\Phi_2 \circ h \circ \Phi_1 = h \circ \Phi_1$, from which $(h_2 \circ \Phi_2) \circ (h_1 \circ \Phi_1) = h_2 \circ (\Phi_2 \circ h_1 \circ \Phi_1) = (h_2 \circ h_1) \circ \Phi_1$.

Proof (Thm. 1). It is obvious that $\mathtt{F}$ preserves the identities. For composition, let $f_1 = h_1 \circ \Phi_1 : \Phi_2 \to \Phi_1$ and $f_2 = h_2 \circ \Phi_2 : \Phi_3 \to \Phi_2$ in $\mathtt{Sym}(\mathtt{C})^{op}$. We have $\mathtt{F}(f_1 \circ f_2)_n(h' \circ \Phi_3) = \mathtt{F}((h_2 \circ \Phi_2) \circ (h_1 \circ \Phi_1))_n(h' \circ \Phi_3) = \mathtt{F}(h_2 \circ h_1 \circ \Phi_1)_n(h' \circ \Phi_3) = h' \circ h_2 \circ h_1 \circ \Phi_1 = \mathtt{F}(h_1 \circ \Phi_1)(h' \circ h_2 \circ \Phi_2) = \mathtt{F}(h_1 \circ \Phi_1)_n(\mathtt{F}(h_2 \circ \Phi_2)_n(h' \circ \Phi_3))$. It remains to show that $\mathtt{F}\Phi$ is pullback-preserving. For i ranging over a set I, consider a tuple of arrows $f_i : n_i \to m$ and the corresponding arrows $g_i : n \to n_i$, forming a pullback diagram D in $\mathtt{C}$. We prove that $\mathtt{F}\Phi D$ is a pullback diagram.

Let $k_i : s \to \mathtt{F}\Phi n_i$ be a tuple of functions commuting with the $\mathtt{F}\Phi f_i$. We have to find a unique $u : s \to \mathtt{F}\Phi n$ making the diagram commute. For each $x \in s$, and each index i, let $k_i(x) = h_i \circ \Phi$.

Commutativity of the k_i with the $\mathtt{F}\Phi f_i$ implies that for each i and j, $f_i \circ h_i \circ \Phi = f_j \circ h_j \circ \Phi$. Let $\bar{i} \in I$ a choice of an index, $\rho_{\bar{i}} \in \Phi$ a choice of an isomorphism in Φ, and ρ_i, for $i \in I \setminus \{\bar{i}\}$, such that $f_{\bar{i}} \circ h_{\bar{i}} \circ \rho_{\bar{i}} = f_i \circ h_i \circ \rho_i$. Observe that the tuple $h_i \circ \rho_i$, for $i \in I$, forms a commuting cone of the diagram f_i in $\mathtt{C}$. Hence there exists a unique arrow $v : dom\Phi \to n$ making the diagram commute.

We define $u(x) = v \circ \Phi$. Commutativity is obtained because $g_i \circ v \circ \Phi = h_i \circ \rho_i \circ \Phi = h_i \circ \Phi = k_i(x)$. For uniqueness, let $u' : s \to \mathtt{F}\Phi n$ making the diagram commute. By preservation of monos, for all i, $\mathtt{F}\Phi g_i$ is mono, and $\mathtt{F}\Phi g_i \circ u = \mathtt{F}\Phi g_i \circ u' = k_i$, hence $u = u'$.

The diagrams below depict the case of two arrows.

$$
\begin{tikzcd}
 & m & \\
n_1 \arrow[ur, "f_1"] & n \arrow[l, "g_1"'] \arrow[r, "g_2"] \arrow[d, dashed, "v"] & n_2 \arrow[ul, "f_2"'] \\
 & dom\Phi \arrow[ul, "h_1 \circ \rho_1"] \arrow[ur, "h_2 \circ \rho_2"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathtt{F}\Phi m & \\
\mathtt{F}\Phi n_1 \arrow[ur, "\mathtt{F}\Phi f_1"] & \mathtt{F}\Phi n \arrow[l, "\mathtt{F}\Phi g_1"'] \arrow[r, "\mathtt{F}\Phi g_2"] \arrow[d, dashed, "u"] & \mathtt{F}\Phi n_2 \arrow[ul, "\mathtt{F}\Phi f_2"'] \\
 & s \arrow[ul, "k_1"] \arrow[ur, "k_2"'] &
\end{tikzcd}
$$

Proof (Prop. 2). Injectivity on objects is trivial. Let $f \circ \Phi' \neq g \circ \Phi'$ two arrows in $\mathtt{Sym}(\mathtt{C})^{op}$ from Φ to Φ'. Then the corresponding natural transformations differ at stage $dom(\Phi)$ on the element $id_{dom(\Phi)} \circ \Phi$ because their respective action is $f \circ \Phi' \neq g \circ \Phi'$.

Proof (Prop. 3). We have

$$
\begin{aligned}
& f \in \mathtt{Set}^{\mathtt{C}}_{\Diamond}(\mathtt{F}\Phi, \mathtt{G}_1 + \mathtt{G}_2) \\
\iff\ & f_n : \mathtt{F}\Phi n \to \mathtt{G}_1 n + \mathtt{G}_2 n \\
\iff\ & f_n : \{h \circ \Phi \mid h \in \mathtt{C}(dom(\Phi), n)\} \to \{\langle 1, x\rangle \mid x \in \mathtt{G}_1 n\} \cup \{\langle 2, x\rangle \mid x \in \mathtt{G}_2 n\}
\end{aligned}
$$

Let $f_{dom(\Phi)}(id \circ \Phi) = \langle i, y\rangle$, for $i \in \{1, 2\}$ (the index of the coproduct). Then by naturality, for all n, we have $f_n(h \circ \Phi) = (\mathtt{G}_1 h + \mathtt{G}_2 h)(f_{dom(\Phi)}(id \circ \Phi)) = \langle i, \mathtt{G}_i(y)\rangle$, that is, all the images of the component functions f_n are a subset of the same index of the coproduct at stage n. Hence, we have $\mathtt{Set}^{\mathtt{C}}_{\Diamond}(\mathtt{F}\Phi, \mathtt{G}_1 + \mathtt{G}_2) = \mathtt{Set}^{\mathtt{C}}_{\Diamond}(\mathtt{F}\Phi, \mathtt{G}_1) + \mathtt{Set}^{\mathtt{C}}_{\Diamond}(\mathtt{F}\Phi, \mathtt{G}_2)$.

Proof (Lemma 2). Let $st(x) \xleftarrow{f_x} s \xrightarrow{f_y} st(y)$ be a canonical choice of a span between the two stages of x and y under the hypothesis of Def. 9. We have $f_x \in D^x$ and $f_y \in D^y$, hence we have two arrows $f'_x : n \to s$ and $f'_y : m \to s$ coming from the pullbacks. Moreover, $f_y \circ f'_x$ is in D^y, and $f_x \circ f'_y$ is in D^x (because $\mathtt{G}$ preserves pullbacks, which are then computed in $\mathtt{Set}$). Hence, there are two arrows $\iota_{x \to y} : n \to m$ and $\iota_{y \to x} : m \to n$, again coming from the pullbacks, both making a commuting triangle with f'_x and f'_y. They are easily seen to be a pair of inverses, as all the arrows in $\mathtt{C}$ are monos.

Proof (Lemma 3). Preservation of pullbacks implies that $\mathtt{G}\mathcal{S}_x$ is a pullback object of $\mathtt{G}(D^x)$ in $\mathtt{Set}$, from which existence. For uniqueness, observe that preservation of pullbacks implies preservation of monos, which are injective functions in $\mathtt{Set}$.

Proof (Lemma 4). The first part of the lemma comes from $\mathtt{G}h\widehat{x} \in \mathcal{O}_x$, because of the span $st(x) \xleftarrow{\mathcal{N}_x} st(\widehat{x}) \xrightarrow{h} st(\mathtt{G}h\widehat{x})$. Notice that $h \in D^{\mathtt{G}h\widehat{x}}$, and $st(\widehat{x})$ is the pullback object of the diagram, hence there is an arrow $\rho : st(x) \to st(x)$ such that $\mathcal{N}_{\mathtt{G}h\widehat{x}} = h \circ \rho$, and we have $G\rho\widehat{x} = \widehat{x}$. By hypothesis, ρ is an isomorphism, hence it is in $\mathcal{G}_{\widehat{x}}$.

Proof (Thm. 2). The natural isomorphism $k : \mathtt{G} \to \mathtt{Presh}(\mathtt{SymDec}(\mathtt{G}))$ is defined as

$$k_n(x) = \langle \widehat{x}, \mathcal{N}_x \circ \mathcal{G}_{\widehat{x}}\rangle$$

To see that it is injective, consider $k_n(x) = \langle \widehat{x}, \mathcal{N}_x \circ \mathcal{G}_{\widehat{x}} \rangle = \langle \widehat{y}, \mathcal{N}_y \circ \mathcal{G}_{\widehat{y}} \rangle = k_n(y)$. We have $\widehat{x} = \widehat{y}$ and consequently $\mathcal{N}_x \circ \mathcal{G}_{\widehat{x}} = \mathcal{N}_y \circ \mathcal{G}_{\widehat{x}}$, thus there is $\rho \in \mathcal{G}_{\widehat{x}}$ such that $\mathcal{N}_x = \mathcal{N}_y \circ \rho$. Then $\mathtt{G}\mathcal{N}_x\widehat{x} = \mathtt{G}(\mathcal{N}_y \circ \rho)\widehat{x}$, and $\mathtt{G}\rho\widehat{x} = \widehat{x} \Rightarrow \mathtt{G}\mathcal{N}_x x = \mathtt{G}\mathcal{N}_y y \Rightarrow x = y$.

Next we prove that k_n is surjective. Consider an element $\langle \widehat{x}, h \circ \mathcal{G}_{\widehat{x}} \rangle$ of $\mathtt{Presh}(\mathtt{SymDec}(\mathtt{G}))$. As $\mathtt{G}hx \in \mathcal{O}_x$ by the span $st(x) \xleftarrow{\mathcal{N}_x} st(\widehat{x}) \xrightarrow{h} st(\mathtt{G}h\widehat{x})$, we have $\widehat{x} = \widehat{\mathtt{G}h\widehat{x}} \Rightarrow k_n(\mathtt{G}h\widehat{x}) = \langle \widehat{\mathtt{G}h\widehat{x}}, \mathcal{N}_{\mathtt{G}h\widehat{x}} \circ \mathcal{G}_{\widehat{\mathtt{G}h\widehat{x}}} \rangle = \langle \widehat{x}, \mathcal{N}_{\mathtt{G}h\widehat{x}} \circ \mathcal{G}_{\widehat{x}} \rangle$. From Lemma 4 we also have $\mathcal{N}_{\mathtt{G}h\widehat{x}} \in h \circ \mathcal{G}_{\widehat{x}}$, from which the thesis.

Finally, we show that k is natural. Let $f \in C(n, m)$, and $x \in \mathtt{G}n$. We have $k_m(\mathtt{G}fx) = k_m(\mathtt{G}(f \circ \mathcal{N}_x)\widehat{x}) = \langle \widehat{x}, f \circ \mathcal{N}_x \circ \mathcal{G}_{\widehat{x}} \rangle = \mathtt{Presh}(\mathtt{SymDec}(\mathtt{G}))(f)(\langle \widehat{x}, \mathcal{N}_x \circ \mathcal{G}_{\widehat{x}} \rangle) = \mathtt{Presh}(\mathtt{SymDec}(\mathtt{G}))(f)(k_n(x))$.

Proof (Thm. 4). Consider a tuple of objects $n_{i \in I}$ in $\mathtt{C}$ for I a set. For each cospan $f_i : n_i \to c$ between them, we attempt to construct its unique canonical representative, that is, another cospan $f'_i : n_i \to c'$ having a unique arrow $u : c' \to c$ making the obtained diagram commute. The set of all the canonical representatives then forms the multi-coproduct. Consider all cospans $g_{i \in I, j \in J} : n_i \to c_j$, for J a set, such that there exist at least one $v : c_j \to c$ commuting with the f_i. Intuitively, these cospans are all equivalent because they can be reached by the arrow v in their category of cospans. All the v form a tuple of arrows D having c as a common target. Let m denote the pullback of D. For each $i \in I$, we have a cone $g_{i, j \in J} : n_i \to c^j$ commuting with D, hence we have a unique arrow $\iota_i : n_i \to m$. The cospan $\iota_{i \in I}$ is "almost" the canonical representative of f_i: let d denote the diagonal of the pullback; observe that it is a good candidate as the unique mediating arrow of the multi-coproduct in $\mathtt{C}$, but it does not need to be unique. Any other commuting arrow d' is an arrow in D, hence there is an isomorphism $\rho : m \to m$ such that $d' \circ \rho = d$. Let Φ denote the set of all the isomorphisms ρ obtained in this way; this defines a multi-coproduct in $\mathtt{Sym}(\mathtt{C})$, by giving the canonical representative of the cospan $f_i \circ \Phi_i$ in $\mathtt{Sym}(\mathtt{C})$, where Φ_i is an object of $\mathtt{Sym}(\mathtt{C})$ such that $dom(\Phi_i) = n_i$. The canonical representative is the cospan $\iota_i \circ \Phi_i : \Phi_i \to \Phi$, and the unique mediating arrow is $d \circ \Phi$, that is, the set of all the arrows d' mentioned above.

Proof (Thm. 5). We have to show that the pairing commutes with the projections and is unique. Commutativity is a simple calculation. For uniqueness, observe that the first two components of $h(k)$ are determined by the copairing in $\mathtt{Set}$ of f and g, while the unique choice of the third component comes from uniqueness of the multi-coproduct.

Proof (Prop. 6). For the "only if" direction, let $y = f_n(h \circ \Phi_i) = f_n(\rho \circ h \circ \Phi_i)$ the image of both the elements in the final coalgebra at stage n. Then $h \circ \mathcal{H}_i^f$ contains the normalising arrow of y. By naturality, ρ acts as the identity on y, thus any arrow $\rho \circ h' \in \rho \circ h \circ \mathcal{H}_i^g$ is in D^y. Let $h'' \in h \circ \mathcal{H}_i^g$ be the diagonal of the pullback of D^y. In the pullback diagram of D^y, there is a morphisms (necessarily an isomorphism) ρ' such that $\rho \circ h' \circ \rho' = h''$. Observe that $\rho' \in \mathcal{G}_{\widehat{y}}$. We have $h \circ \mathcal{H}_i^g = h'' \circ \mathcal{G}_{\widehat{y}} = \rho \circ h' \circ \rho' \circ \mathcal{G}_{\widehat{y}} = \rho \circ h' \circ \mathcal{G}_{\widehat{y}} = \rho \circ h \circ \mathcal{H}_i^g$, and by uniqueness of F, we also have $\widehat{y} = g(i)$ from which the thesis.

For the "if" direction, assume that we have an operation ρ such that $\rho \circ h \circ \mathcal{H}_i^g = h \circ \mathcal{H}_i^g$. Then for each $h' \in \mathcal{H}_i^g$ we have an operation $\rho' \in \Phi'_{g(i)}$ (possibly the identity) such that $\rho \circ h \circ h' \circ \rho' = h \circ h'$. This implies that ρ acts as the identity on $f_n(h \circ \Phi_i)$. The thesis then follows by naturality of f.