Families: an *efficient* categorical model of computation with resources

Vincenzo Ciancia

Universidad Complutense de Madrid
and
Institute for Logic, Language and Computation - Amsterdam

joint work with Alexander Kurz and Ugo Montanari

An infinite state system

```
enc_server ( key, chan ):

   currentKey = key

   while true
      if (timeToChangeKey())
             newKey = genKey()
             enc_send(ch,currentKey,newKey)
             currentKey = newKey

      else ...
```

Nominal calculi

- Novel features of programming languages, such as mobility of network components, required new programming paradigms and new models and led to the definition of the π-calculus
- Exploiting a *name generation* construct, reflected by a *scope extrusion* mechanism in the semantics, the π-calculus achieves mobility
- Nowadays, name generation and name passing have proved fundamental: e.g. *sessions* in service oriented computing, *objects* in object-oriented programming, *keys* and *nonces* in security protocols
- Plain old α-conversion!

Beyond names

New programming paradigms stem from **service-oriented computing**, **distributed algorithms**, **peer to peer systems** and so on. They require more than pure names to be modelled.

Some examples:

- **Code mobility:** processes operate in **restricted scopes** and executable code can **move** from a scope to another
  The Ambient Calculus [Cardelli, Gordon]

- **Fusions of names**: different entities may be equated at run-time
  the Fusion Calculus [Parrow, Victor]

- **Multi-layered networks**: entire sub-networks (darknets) may be *hidden*, and their existence may be discovered and communicated
  Network Coordination Policies [Ciancia, Ferrari, Guanciale, Strollo]

Modelling and analysis

- First challenge (well known, elegant solutions):
  The definition of bisimulation in these calculi is non-standard. Can we provide a standard, fully abstract denotational model?
  Coalgebras over Presheaves [Too many to be cited!]

- Second challenge (work in progress):
  provide finite (garbage collecting) representations
  pure names: **History-dependent automata [Montanari, Pistore]**
  the general case: **Families**

- Third challenge (future work):
  implement analysis algorithms

  model-checking, minimisation, trace analysis, monitoring, all of them need finite models, and **garbage collecting** semantics.

  Minimization for the case of names already implemented **[Emilio Tuosto's thesis]**

Presheaves

- Presheaves can be seen as a structued, multi-sorted generalisations of set theory
- Formally, **functor categories $\mathbf{Set}^{\mathbf{C}}$** from a small category $\mathbf{C}$ to $\mathbf{Set}$
- Why presheaves? **Coalgebras over presheaves** give a standard framework to define the abstract semantics of modern languages
- and also to define suitable coalgebraic logics **[Bonsangue, Kurz - LICS 2007]**

The case of names

- Modelling names: $\mathbf{Set}^{\mathbf{I}}$ where $\mathbf{I}$ is the category of *finite subsets of the natural numbers* and **injections**
- Let P :$\mathbf{Set}^{\mathbf{I}}$. Then $x \in P(n)$ roughly means that the free names of x are included in n.
- The full subcategory of $\mathbf{Set}^{\mathbf{I}}$ consisting of functors that preserve pullbacks is called the Schanuel topos

Pullback preservation

- Recall that arrows in **I** are monos. Preserving pullbacks means **preserving intersections**
  "if the free names of x are in n and in m then they are in $n \cap m$"
- Existence of a minimal n where an element is found: the **support**
- **[Fiore, Staton - IC 2006]**
  [Gadducci, Miculan, Montanari - HOSC 2006]

  Categorical equivalence between
  - the **nominal sets** of Gabbay and Pitts
  - the **Schanuel topos**
  - the **named sets** of Montanari and Pistore (whose coalgebras are HD-automata)

Global names

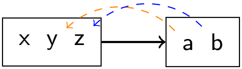
- The world wide web
- There is a global naming autority
- Just knowing the name of an entity is enough to connect to it
- Consequence: the newly generated entities are all different, that is, **no garbage collection**

Local names

- Local area networks, peer-to-peer systems, darknets...
- There is no global naming authority
- To reach an entity one needs to have network connectivity to it, and a designated name (in the local name space) to refer to it
- If an entity is new, it is new. All the newly generated names are in principle equal
- **From global to local names: quotient with respects to all relabellings**

Named Sets

- Basic idea: model states of a system as a set whose elements have a set of "local names", or placeholders, attached
- Morphisms trace the *history* of names using an associated injection from names of the destination to names of the source



- The simple description above is not enough: we also need to quotient these relabellings with symmetries over names.

From named sets to families

1. named sets are a concrete description of **free coproduct completions** of finite permutation groups and quotients of sets of injective functions: they are **families** over that category
[in Sam Staton's thesis]

2. Locality of names and garbage collection arise from the definition of various functors over named sets, by a careful choice over all the possible isomorphic definitions trough the equivalence with presheaves
[in my thesis]

3. families over groups of isomorphisms of a generic index category are **an alternative to presheaves**, where interfaces are *local*. Locality of interfaces and garbage collection as in (2).
[This work]

4. As a bonus, coalgebraic minimisation can be implemented, and gives rise to **symmetry reduction** and **elimination of redundant resources**

Families

Given a **small** category **C**, we define the category **Fam**(**C**)

- Objects are coproducts in **Set**: indexed collections of objects of **C**

$$\coprod_{i \in I} \{c_i\}$$

- Each $i \in I$ is considered an **element** whose **local interface** is c_i
- An arrow from $\coprod_{i \in I}\{c_i\}$ to $\coprod_{j \in J}\{d_j\}$ is a function $h : i \to j$ and a family of **C** arrows

$$\coprod_{i \in I} \{\mathcal{H}_i : c_i \to d_{h(i)}\}$$

Working with families

- Idea: use families for the abstract semantics of programming languages directly
- Not so convenient: due to locality of interfaces, the specification of functors and coalgebras gets very technical
- Presheaves are "like sets" and specifications are elegant
- **An equivalence of categories** lifts to equivalent functors, equivalent categories of algebras and coalgebras.
- Therefore: we may use **presheaves as a specification language**, and families as a **correcty by construction** model for implementation

Question

Q: what categories of presheaves can be represented as families?

1. Our answer: **small index categories of monos,** all automorphisms are iso, and **(weak) wide pullback preservation** give rise to an **equivalence of categories**

2. **[Adamek, Velebil - TAC 2008]**: **locally presentable index categories** and **weak wide pullback preservation** represent presheaves - natural transformations are not encoded.
Generalises Joyal's species as representations of analytic functors.

- The two conditions are different: (1) includes coproducts of categories, (2) includes Set
They obviously overlap (e.g. finite sets and injections).

Towards a general theory of *symmetrised* representation

- The index categories of presheaves give us an uniform “alphabet” to model different kinds of resources. Some examples:
  1. Disjoint union of different resource types (coproduct of categories)
  2. Explicit fusions (equivalence classes and **monic** maps)
  3. Fusions [Miculan - MFPS 2008] (finite sets and functions between them)
  4. Distinctions [Ghani, Yemane, Victor - CMCS 2004] (irreflexive finite relations and all relation-preserving functions) for the open pi-calculus
- (1,2) fall under our hypothesis. Saturation gives infinitary systems - not good for minimization. Future work: saturate w.r.t. minimal contexts.
- (3,4) are not of monos, and lack wide pullbacks in the index category.

Support and symmetry

- We represent the wide pullback preserving full subcategory of $\mathbf{Set}^{\mathbf{C}}$ as the category $\mathbf{Fam}(\mathbf{Sym}(\mathbf{C})^{op})$
- $\mathbf{Sym}(\mathbf{C})$ is the category of groups of automorphisms of $\mathbf{C}$, representing the **support** and **symmetry** of an element of a presheaf (more in the following)
- Symmetries are the essential information that is needed to reconstruct each represented presheaf: first one reconstructs the presheaf "freely" using representables, then a quotient is made using the symmetry.

Support, symmetry, orbit

Let P be a wide-pullback preserving presheaf in $\mathbf{Set}^{\mathbf{C}}$

- The **support** $supp(x)$ of $x \in P(n)$ is the least m such that $P(f : m \to n)(y) = x$
- **Orbits** are connected components in the **category of elements**, that is, elements that can be obtained from each other by the action of P on morphisms
- The **representative** x^o of (the orbit of) x is its minimal element (if it exists!)
- The **symmetry** $\mathcal{G}(x)$ is the set of isomorphisms $\phi : supp(x) \to supp(x)$ such that $P(\phi)(x^o) = x^o$

The product: local interfaces

- The product in the case of presheaves is "point-wise": just pairs!
- In families, we have to establish a mapping of interfaces and there are plenty of them

$$(\bar{x}y.0 \parallel \bar{y}x.0) \xrightarrow{\{x \mapsto x, y \mapsto y\}} \{x, y, a, b\} \xleftarrow{\{x \mapsto a, y \mapsto b\}} (\bar{x}y.0 \parallel \bar{y}x.0)$$

$$=$$

$$\left\langle (\bar{x}y.0 \parallel \bar{y}x.0), (\bar{a}b.0 \parallel \bar{b}a.0) \right\rangle$$

$$(\bar{x}y.0 \parallel \bar{y}x.0) \xrightarrow{\{x \mapsto a, y \mapsto b\}} \{a, b, c\} \xleftarrow{\{x \mapsto b, y \mapsto c\}} (\bar{x}y.0 \parallel \bar{y}x.0)$$

$$=$$

$$\left\langle (\bar{a}b.0 \parallel \bar{b}a.0), (\bar{b}c.0 \parallel \bar{c}b.0) \right\rangle$$

Name mappings as cospans

- To represent name mappings we use *cospans*, that is pairs of mappings with a common target
- In the general case, these must keep in account symmetries, so they are not injections, but rather arrows in a suitable category of symmetries. In the simplest case, they are injections.
- The target object must be **minimal** and **uniquely determined**: names that are not in the image of at least one arrow are not interesting
- The multi-coproduct $MCP(c, d)$ [Diers, 1979] is a categorical construction that identifies all isomorphic cospans, and only find the minimal ones

The categorical product

- The categorical product of $\coprod_{i \in I}\{c_i\}$ and $\coprod_{j \in J}\{d_j\}$ is a family

$$\coprod_{\langle i \in I, j \in J, c \in MCP(c_i, d_j) \rangle} e_{\langle i,j,c \rangle}$$

  where $e_{\langle i,j,c \rangle}$ is the target of the cospan c

- That is, we take pairs of elements, and a binding of their interfaces into a bigger, common one.

Consequences on bisimulation and model checking

The definition of the product has consequences on the definition of the bisimulation problem and the model checking problem

- **Bisimulation** becomes a ternary relation, employing two states and a name mapping between them.
- The **satisfaction relation** between states and (modal) formulas is a ternary relation in turn. Names that are not mapped are not involved in the particular instance of the satisfaction definition
- **New algorithms have to be invented to efficiently handle these ternary relations** in minimisation and bisimulation checking. Complicated by the presence of symmetries!

Symmetry reduction

- The symmetry **always grows along morphisms**, including coalgebra homomorphisms
- The symmetry in the final coalgebra is the greatest possible symmetry up-to bisimulation, or **behavioural symmetry**
- The unique morphism in the final coalgebra gives the common interface and symmetry of each class of bisimilar elements

Iteration along the terminal sequence

- The final coalgebra gives us the **active subobject of the interface of an element** x**,** and its **behavioural symmetry**
- Both may be computed by iteration along the terminal sequence
- Implemented for the case of names in **Ferrari, Montanari, Tuosto** - **TCS 2005 (MIHDA - MInimizing HD-Automata)**
- The minimal system is a compressed representation that exploits the symmetry. Useful for model-checking [Emerson et al.].

Conclusions and future work

- The **species** of Joyal, the **families** of Diers, the theory of presheaves for the semantics of programming languages, and the theory of history-dependent automata

- All these ingredients, combined together, give rise to a *garbage-collecting* model for the semantics of programming languages, which sits side by side to presheaves

- Presheaves are good for the specification of the semantics, families for its implementation

- Ongoing work: go beyond categories of monos

- Future work: coalgebraic logics (with global or local interfaces???) and model checking algorithms

Support, symmetry, orbit (reprise)

Let P be a wide-pullback preserving presheaf in $\mathbf{Set}^{\mathbf{C}}$

- The **support** $supp(x)$ of $x \in P(n)$ is the least m such that $P(f : m \to n)(y) = x$
- **Orbits** are connected components in the **category of elements**, that is, elements that can be obtained by each other by the action of morphisms
- The **representative** x^o of (the orbit of) x is its minimal element (if it exists!)
- The **symmetry** $\mathcal{G}(x)$ is the set of isomorphisms $\phi : supp(x) \to supp(x)$ such that $P(\phi)(x^o) = x^o$

Coproducts of symmetrised representables

- Recall that the category of elements determines the presheaf: **presheaves are colimits of representables** where the diagram is determined by the category of elements

- In the special case we study, they are **coproducts of symmetrised representables** (see also [Adamek, Velebil - TAC 2008])

$$\coprod_{x^o} Hom(supp(x^o), -)/_{\mathcal{G}(x^o)}$$

- The above representation is concisely described by a family in $\mathbf{Sym}(\mathbf{C}^{op})$

$$\coprod_{x^o} \{\mathcal{G}(x^o)\}$$

Morphisms and locality of interfaces

- We denote presheaves by **representatives of orbits**
- But a natural transformation **does not** map representatives to representatives!
- However, we can reconstruct a natural transformation $f : P \longrightarrow P'$ by recording, for each representative x with support n, the representative y of $f_n(x)$, and an arrow $\mathcal{H}$ such that $P(\mathcal{H})(y) = f_n(x)$
- This gives rise to an arrow in $\mathbf{Fam}(\mathbf{Sym}(\mathbf{C})^{op})$. The arrow $\mathcal{H}$ is the **history** of the interface of x along the morphism f.
- The definition on arrows completes an equivalence of categories