On endofunctors modelling higher-order behaviours

Marco Peressotti

Department of Mathematics and Computer Science, University of Udine, Italy
marco.peressotti@uniud.it

It is well known that *higher-order systems*, i.e. systems which can pass around systems of the same kind, like the λ-calculus [1,3], CHOCS [18], the higher-order π-calculus [14], HOcore [10], etc., are difficult to reason about. Many bisimulations and proof methods have been proposed also in recent works [4, 9, 11, 12, 15–17]. This effort points out that a definition of *abstract* higher-order behaviour is still elusive. In this work, we show how these abstract behaviours can be modeled as the *final coalgebras* of suitable *higher-order behavioural functors*.

Coalgebras are a well established framework for modelling and studying concurrent and reactive systems [13]. In this approach, we first define a *behavioural endofunctor* B over Set (or other suitable category), modelling the computational aspect under scrutiny; for X a set of states, BX is the type of behaviours over A. Then, a system over A corresponds to a B-coalgebra, i.e. a map $\alpha : X \to BX$ associating each state with its behaviour. The crucial step of this approach is defining the functor B, as it corresponds to specify the behaviours that the systems are meant to exhibit. Once we have defined a behavioural functor, many important properties and general results can be readily instantiated, such as the existence of the *final B-coalgebra* (containing all abstract behaviours), the definition of the canonical *coalgebraic bisimulation* (which is the abstract generalization of Milner's strong bisimilarity) and its coincidence with behavioural equivalence [2], the construction of canonical *trace semantics* [8] and *weak bisimulations* [5], the notion of *abstract GSOS* [19], etc. We stress the fact that behavioural functors are *syntax agnostic*: they define the semantic behaviours, abstracting from any particular concrete representation of the systems.

Despite these results, a general coalgebraic treatment of higher-order systems is still missing. In fact, defining these functors for higher-order behaviours is challenging. In order to describe the problem, let us consider first a functor for representing the behaviour of a first-order calculus, like CCS with value passing:

$$B : \mathsf{Set} \to \mathsf{Set} \qquad BX = \mathcal{P}_f(L \times V \times X + L \times X^V + X) \tag{1}$$

where L is the set of labels and V is the set of values. This functor is well-defined, and it admits a final coalgebra which we denote by νB; the carrier of this coalgebra is the set of all possible behaviours, i.e., synchronization trees labelled with nothing, input or output actions. Now, in a higher-order calculus like HOπ, the values that processes can communicate are processes themselves. Semantically, this means that a higher-order behaviour can communicate *behaviours*; hence, in the definition (1) we should replace V with the carrier of νB, as follows:

$$B_{ho} : \mathsf{Set} \to \mathsf{Set} \qquad B_{ho}X = \mathcal{P}_f(L \times |\nu B_{ho}| \times X + L \times X^{|\nu B_{ho}|} + X) \tag{2}$$

But this means that we are defining B_{ho} using its own final coalgebra νB_{ho}, which can be defined (if it exists) only after B_{ho} is defined—a circularity!

We think that this circularity is the gist of higher-order behaviours: any attempt to escape it would be restricting and distorting. One may be tempted to take as V some (syntactic) representation of behaviours (e.g., processes), but this would fall short. First, the resulting behaviours would not be really higher-order, but rather behaviours manipulating some *ad hoc* representation of behaviours. Secondly, we would need some mechanism for moving between behaviours and their representations–which would hardly be complete. Third, the resulting functor would not be abstract and independent from the syntax of processes, thus hindering the possibility of reasoning about the computational aspect on its own, and comparing different models sharing the same kind of behaviour.

Endofunctors describing behaviours with input and outputs as (1) can be seen as endofunctors with *mixed-variance parameters*, e.g. $F\colon \mathsf{Set}^{op} \times \mathsf{Set} \to [\mathsf{Set}, \mathsf{Set}]$ where $F(A,B) = Id^A + B$. Since we are interested in endofunctors with a final coalgebra we shall consider "schemes of endofunctors" in some suitable subcategory E of $[\mathsf{C}, \mathsf{C}]$. Actually, parameters do not have to be in C, for instance, the functor $Id^A\colon \mathsf{Top} \to \mathsf{Top}$ can be seen as parametric in the exponentiable space $A \in \mathsf{ExpTop}$. In this situation, we need some coherent way back to the category of parameters i.e. a functor from E to D. An example of such situation is offered by taking E to be the category of biconinuous endofunctors over Set, as shown in [?]: every such functor admits a final coalgebra whose carrier set is endowed with a complete partial order naturally induced by the final sequence. Therefore, given:

$$F\colon \mathsf{D}^{op} \times \mathsf{D} \to \mathsf{E} \quad \text{and} \quad N\colon \mathsf{E} \to \mathsf{D}$$

(the driving example being $N = |\nu -|$) we are interested in finding $B : \mathsf{C} \to \mathsf{C} \in \mathsf{E}$ s.t.:

$$B \cong F(NB, NB)$$

or, equivalently, $Z \in \mathsf{C}$ s.t.:

$$Z \cong NF(Z, Z)$$

since $B \cong F(Z, Z)$ and $Z \cong NB$.

A $\omega\mathsf{Cat}$-*category* is a 2-category whose hom-categories have colimits for all ω-chains and composition preserves them. A $\omega\mathsf{Cat}_0$-category is a $\omega\mathsf{Cat}$-category whose hom-categories have initial objects and composition preserves them. Any category enriched over Cpo, the category of continuous maps between complete partial orders, such as Cpo itself is an $\omega\mathsf{Cat}$-category. Likewise, any category enriched over $\mathsf{Cpo}_\perp$ is an $\omega\mathsf{Cat}_0$-category.

Theorem 1. *Assume D and E to be $\omega\mathsf{Cat}_0$-categories with pseudo initial objects and pseudo colimits of ω-chains of coreflections. Assume F and N to be a pseudo $\omega\mathsf{Cat}$-functors. There exist $G : \mathsf{C} \to \mathsf{C} \in \mathsf{E}$ and $Z \in \mathsf{C}$ as above.*

Example 2 (Higher-order deterministic processes). Let C and D be $\mathsf{Cpo}_\perp$ and consider the parametric family of endofunctors $Id^A + Id + B$. The components of the coproduct model inputs, internal moves, and termination with outputs, respectively. In the higher-order version of this behaviour inputs and outputs are behaviours of the same kind. If we set aside for a moment syntax and binders (which can be modelled in suitable presheaf categories [6,7]), this behaviour offers an operational semantics for the λ-calculus: intuitively, inputs are transitions $t \xrightarrow{z} t\, z$ whereas internal reductions and outputs are transitions $(\lambda x.t)\, z \to t[x/z]$.

The functor $F(A,B) = Id^A + Id + B$ is Cpo-enriched and each endofunctors in its image has a final coalgebra in $\mathsf{Cpo}_\perp$. By restricting to the image of F, the assignment $|\nu -|$ defines a Cpo-enriched functor $N\colon \mathsf{E} \to \mathsf{Cpo}_\perp$. Thus, we have $B_\lambda\colon \mathsf{Cpo}_\perp \to \mathsf{Cpo}_\perp$ and $Z_\lambda \in \mathsf{Cpo}_\perp$ s.t.:

$$B_\lambda \cong Id^{Z_\lambda} + Id + Z_\lambda \quad \text{and} \quad Z_\lambda \cong |\nu B_\lambda|.$$

A B_λ-coalgebra $(X \to X^{Z_\lambda} + X + Z_\lambda)$ is a strict continuous map assigning to each state of its carrier *(a)* a strict continuous function assigning a continuation to any value in input, *(b)* or a new state (internal step), *(c)* or a value (output). Indeed values are elements of the $CPO_\perp$ Z_λ carrying the final coalgebra of B_λ i.e. behaviours for B_λ itself.

Finite-order approximations It might not be so easy to work with the solution $B \cong F(N^{op}(B), N(B))$ since it is defined in terms of its own final coalgebra. The reason of this is rooted in the inherent circularity of higher-order definitions; circularity that resurfaces in the definition of B and many related constructions such as B-bisimulations.

The steps in the computation of the fixed point B are finite-order behaviours B_n (and Z_n):

$$Z_0 = 0_{\mathsf{D}} \qquad Z_n = N(B_n) \qquad B_0 = 0_{\mathsf{E}} \qquad B_{n+1} = F(Z_n, Z_n)$$

approximating B (for it is the (co)limit of the resulting ω-chain of coreflections in E). Therefore, B-bisimulations may be given by induction on n deriving B_{n+1}-bisimulations from projections to n-order behaviours and B_n-bisimulations. Embeddings guarantee coherence.

References

[1] S. Abramsky. The lazy lambda calculus. *Research topics in functional programming*, pages 65–116, 1990.

[2] P. Aczel and N. Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Proc. CTCS*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365, 1989. Springer.

[3] H. Barendregt. *The lambda calculus: its syntax and its semantics.* Studies in Logic and the Foundations of Mathematics. North-Holland, 1984.

[4] L. Birkedal, R. E. Møgelberg, J. Schwinghammer, and K. Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Logical Methods in Computer Science*, 8(4), 2012.

[5] T. Brengos, M. Miculan, and M. Peressotti. Behavioural equivalences for coalgebras with unobservable moves. *CoRR*, abs/1411.0090, 2014.

[6] M. Fiore and D. Turi. Semantics of name and value passing. In H. Mairson, editor, *Proc. 16th LICS*, pages 93–104, Boston, USA, 2001. IEEE Computer Society Press.

[7] M. P. Fiore, G. D. Plotkin, and D. Turi. Abstract syntax and variable binding. In G. Longo, editor, *Proc. 14th LICS*, pages 193–202, 1999. IEEE Computer Society Press.

[8] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

[9] V. Koutavas, P. Blain Levy, and E. Sumii. From applicative to environmental bisimulation. In *Proc. MFPS*, *Electronic Notes in Theoretical Computer Science*, 276(0):215–235, 2011.

[10] I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. In *Proc. LICS*, pages 145–155. IEEE Computer Society, 2008.

[11] S. Lenglet, A. Schmitt, and J. B. Stefani. Characterizing contextual equivalence in calculi with passivation. *Information and Computation*, 209(11):1390–1433, 2011.

[12] A. Piérard and E. Sumii. A higher-order distributed calculus with name creation. In *Proc. LICS*, pages 531–540. IEEE Computer Society, 2012.

[13] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[14] D. Sangiorgi. Bisimulation for higher-order process calculi. *Information and Computation*, 131(2):141–178, 1996.

[15] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *Proc. LICS*, pages 293–302. IEEE Computer Society, 2007.

[16] K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In *Semantics and algebraic specification*, pages 329–375. Springer, 2009.

[17] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM (JACM)*, 54(5):26, 2007.

[18] B. Thomsen. Plain CHOCS: a second generation calculus for higher order processes. *Acta informatica*, 30(1):1–59, 1993.

[19] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS*, pages 280–291. IEEE Computer Society Press, 1997.