Proceedings (preliminary version)

10th International Workshop on

Coalgebraic Methods in Computer Science

Paphos, Cyprus

March 26–28, 2010

Editors:

B.P.F. Jacobs
J.J.M.M Rutten
M. Niqui
A. Silva

Editors:
Bart Jacobs
Radboud University Nijmegen
bart@cs.ru.nl
Milad Niqui
Centrum Wiskunde & Informatica
M.Niqui@cwi.nl
Jan Rutten
Centrum Wiskunde & Informatica and Radboud University Nijmegen
Jan.Rutten@cwi.nl
Alexandra Silva
Centrum Wiskunde & Informatica
A.M.Silva@cwi.nl

Typeset with LaTeX 2$_\varepsilon$
Printed in Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
Cover design: Jos van der Werf
Cover photograph: 'Sunset at Aphrodite's Rocks' © Copyright 2007 Dan Barbus

Note: This is the preliminary version of the proceedings. For citation purposes please use the final version that will be published by Elsevier.

Contents

Preface

This volume contains the proceedings of the Tenth Workshop on Coalgebraic Methods in Computer Science (CMCS'10). The workshop was held in Paphos, Cyprus from March 26 until March 28, 2010, as a satellite event to the European Joint Conference on Theory and Practice of Software (ETAPS'10). The aim of the CMCS workshop series is to bring together researchers with a common interest in the theory of coalgebras and its applications.

Coalgebras have been found extremely useful for capturing state-based dynamical systems, such as transition systems, automata, process calculi, and class-based systems. The theory of coalgebras has developed into a field of its own interest, presenting a deep mathematical foundation and a growing domain of applications and interactions with various other fields, such as reactive and interactive systems theory, object oriented and concurrent programming, formal system specification, modal logic, dynamical systems, control systems, category theory, algebra, and analysis.

Previous workshops have been organised in Lisbon (1998), Amsterdam (1999), Berlin (2000), Genoa (2001), Grenoble (2002) Warsaw (2003), Barcelona (2004), Vienna (2006) and Budapest (2008). Starting from 2004 CMCS has become biennial, alternating with CALCO (Conference on Algebra and Coalgebra in Computer Science), which, for odd-numbered years, has been formed by the union of CMCS with WADT (Workshop on Algebraic Development Techniques).

In 2010, the 10th edition of CMCS is celebrated. For this special occasion all members of previous CMCS program committees have been invited to be a member in 2010. Additionally, four specialists in the field have been invited to present overviews of both obtained results and future challenges in important subareas:

- Venanzio Capretta: Coalgebra in functional programming and type theory
- Bartek Klin: Operational semantics coalgebraically
- Dirk Pattinson: Logic and coalgebra
- Ana Sokolova: Probabilistic systems coalgebraically

The Programme Committee of CMCS'10 consisted of: Jiří Adámek (Braunschweig), Alexandru Baltag (Oxford), Luis Barbosa (Braga), Marcello Bonsangue (Leiden), Corina Cîrstea (Southampton), Robin Cockett (Calgary), Andrea Corradini (Pisa), Neil Ghani (Glasgow), Peter Gumm (Marburg), Furio Honsell (Udine), Bart Jacobs (Nijmegen, co-chair), Bartek Klin (Cambridge), Clemens Kupke (London), Alexander Kurz (Leicester), Marina Lenisa (Udine), Stefan Milius (Braunschweig), Ugo Montanari (Pisa), Larry Moss (Bloomington), Milad Niqui (Amsterdam), Dirk Pattinson (London), Dusko Pavlovic (Oxford), John Power (Edinburgh), Horst Reichel (Dresden), Grigore Rosu (Urbana), Jan Rutten (Amsterdam, co-chair), Davide Sangiorgi (Bologna), Lutz Schroeder (Bremen), Alexandra Silva (Amsterdam), Hendrik Tews (Nijmegen),

Tarmo Uustalu (Tallinn), Yde Venema (Amsterdam), Hiroshi Watanabe (Osaka), James Worrell (Oxford).

The papers were refereed by the program committee and by several outside referees, whose help is gratefully acknowledged.

The Organising Committee: *Bart Jacobs (PC chair), Milad Niqui (co-chair), Jan Rutten (PC chair), Alexandra Silva (co-chair).*

Recursive Program Schemes and Context-Free Monads

Jiří Adámek[a] Stefan Milius[a] Jiří Velebil[b,1]

[a] *Institut für Theoretische Informatik, Technische Universität Braunschweig, Germany*

[b] *Faculty of Electrical Engineering, Czech Technical University of Prague, Czech Republic*

Abstract

Solutions of recursive program schemes over a given signature Σ were characterized by Bruno Courcelle as precisely the *context-free* (or *algebraic*) Σ-trees. These are the finite and infinite Σ-trees yielding, via labelling of paths, context-free languages. Our aim is to generalize this to finitary endofunctors H of general categories: we construct a monad C^H "generated" by solutions of recursive program schemes of type H, and prove that this monad is ideal. In case of polynomial endofunctors of Set our construction precisely yields the monad of context-free Σ-trees of Courcelle. Our result builds on a result by N. Ghani et al on solutions of algebraic systems.

Keywords: algebraic trees, recursive program schemes, ideal theory, monads

1 Introduction

The aim of the current paper is to introduce, for a finitary endofunctor H of a "reasonable" category, the context-free monad C^H of H characterizing solutions of recursive program schemes of type H. This is analogous to our previous construction of the rational monad R^H characterizing solutions of first-order recursive equations of type H, see [4]. In case of a polynomial functor $H = H_\Sigma$ on Set the monad R^H is given by all rational Σ-trees, i. e., Σ-tree having (up to isomorphism) only a finite set of subtrees, see [16]. In contrast, C^H is given by the algebraic trees investigated in the pioneering paper of Bruno Courcelle [9]. We call these trees t *context-free* since in [9] they

[1] Supported by the grant MSM 6840770014 of the Ministry of Education of the Czech Republic.

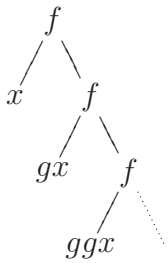
This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

are characterized by the property that a certain natural language associated to the paths of t is context-free (whereas t is rational iff that language is regular).

Recall that a *recursive program scheme* (or rps for short) defines new operations $\varphi_1, \ldots, \varphi_k$ of given arities $n_1, \ldots, n_k$ recursively, using given operations represented by symbols from a signature Σ. Here is an example:

$$\varphi(x) = f(x, \varphi(gx)) \tag{1}$$

is a recursive program scheme defining a unary operation f from the givens in $\Sigma = \{ f, g \}$ with f binary and g unary. The semantics of recursive program schemes is a topic at the heart of theoretical computer science, see [9,17]. Here we are interested in the so-called uninterpreted semantics, which treats a recursive program scheme as a purely syntactic construct, and so its solution is given by Σ-trees over the given variables. For example, the uninterpreted solution for φ above is the Σ-tree



(2)

(here we simply put the terms x, gx, ggx, etc. for the corresponding subtrees).

Observe that if $\Phi = \{ \varphi_1, \ldots, \varphi_k \}$ denotes the signature of the newly defined operations and

$$H_\Phi X = X^{n_1} + \cdots + X^{n_k}$$

is the corresponding polynomial endofunctor of Set, then algebras for H_Φ are just the classical general algebras for the signature Φ. We denote by F^H the free monad on H, thus F^{H_Φ} is the monad of finite Φ-trees. A recursive program scheme can be formalized as a natural transformation

$$e : H_\Phi \to F^{H_\Sigma + H_\Phi}.$$

In fact, $F^{H_\Sigma + H_\Phi}$ is the monad of all finite $(\Sigma + \Phi)$-trees. Since X^{n_i} is a functor representable by n_i, a natural transformation from X^{n_i} into $F^{H_\Sigma + H_\Phi}$ is, by Yoneda Lemma, precisely an element of $F^{H_\Sigma + H_\Phi}(n_i)$, i. e., a finite $(\Sigma + \Phi)$-tree on n_i variables. Thus, to give a natural transformation e as above means precisely to give k equations, one for each operation symbol φ_i from Φ,

$$\varphi_i(x_0, \ldots, x_{n-1}) = t_i \qquad (i = 1, \ldots, k) \tag{3}$$

where t_i is a $(\Sigma + \Phi)$-term on $\{ x_0, \ldots, x_{n-1} \}$. This is the definition of a recursive program scheme used in [9].

An uninterpreted solution of $e : H_\Phi \to F^{H_\Sigma + H_\varphi}$ is a k-tuple of Σ-trees $t_1^\dagger, \ldots, t_k^\dagger$ such that the above formal equations (3) become identities under the simultaneous second-order substitution [2] of t_i for f_i, for $i = 1, \ldots, k$. For example, the tree $t^\dagger(x)$ from (2) satisfies the corresponding equality of trees

$$t^\dagger(x) = g(x, t^\dagger(fx)).$$

This concept of solutions was formalized in [21] by means of the free completely iterative monad T^H on a functor H; in case $H = H_\Sigma$ this is the monad of all Σ-trees. We recall this in Section 2. The uninterpreted solution is a natural transformation $e^\dagger : H_\Phi \to T^{H_\Sigma}$ and this leads us to the following reformulation (and renaming) of the concept of an algebraic tree of Courcelle [9]:

Definition 1.1 A Σ-tree is called *context-free* if there exists a recursive program scheme (3) such that $t = t_1^\dagger$.

Example 1.2 Every rational tree is context-free, and (2) shows a context-free tree that is not rational.

Courcelle proved that the monad C^{H_Σ} of all context-free Σ-trees as a submonad of T^{H_Σ} is iterative in the sense of Calvin Elgot [10]. Furthermore, context-free trees are closed under second-order substitution. The aim of the present paper is a construction of the context-free monad C^H for all finitary endofunctors H of locally finitely presentable categories. We prove that this monad is always ideal, i. e., it can be seen as a coproduct of variables and non-variables—this is a desired property that simplifies working with a monad, see e. g. [21,6,15]. However, at this moment we leave as open problems the proofs that C^H is closed under second-order substitution and it is iterative, in general.

Related work. Our work is based on the pioneering paper by Bruno Courcelle [9]. As we mentioned already, Guessarian [17] presents the classical algebraic semantics of recursive program schemes, for example, their uninterpreted solution as infinite Σ-trees and their interpreted semantics in ordered algebras. The realization that basic properties of Σ-trees stem from the fact that they form the final H_Σ-coalgebra goes back to Larry Moss [22] and also appears independently and almost at the same time in the work of Neil Ghani et al [13] (see also [14]) and Peter Aczel et al [2] (see also [1]). Ghani et al [11] were the first to present a semantics of uninterpreted recursive program schemes in the coalgebraic setting. Their paper contains a solution theorem for uninterpreted

[2] Recall that in general, a simultaneous second-order substitution replaces in a tree over a signature Γ all operation symbols by trees over another signature, Σ, say. See [9] or [21] for a category-theoretic description.

(generalized) recursive program schemes. Here we derive from that the result that all "guarded" recursive program schemes have a unique solution that is a fixed point w. r. t. second-order substitution. The ideas of [11] were taken further in [21]; this fundamental study contains a comprehensive category-theoretic version of algebraic semantics in the coalgebraic setting: the paper provides an uninterpreted as well as interpreted semantics of recursive program schemes and the relation of the two semantics (this is a fundamental theorem in algebraic semantics).

The present paper builds on ideas in [11,21]. Our construction of the context free monad is new. It is inpired by the construction of the rational monad in [4], see also [12] for a more general construction.

Acknowledgements. We are grateful to the anonymous referees for their comments which helped improving the presentation of our results.

2 Construction of the context-free monad

Throughout the paper we assume that a finitary (i. e., filtered colimit preserving) endofunctor H of a category $\mathscr{A}$ is given, and that H preserves monomorphisms. We assume that $\mathscr{A}$ is locally finitely presentable, coproduct injections

$$\mathsf{inl} : X \to X + Y \qquad \text{and} \qquad \mathsf{inr} : Y \to X + Y$$

are always monic, and a coproduct of two monomorphisms is also monic. Recall that local finite presentability means that $\mathscr{A}$ is cocomplete and has a set $\mathscr{A}_{\mathsf{fp}}$ of finitely presentable objects (meaning those whose hom-functor is finitary) such that $\mathscr{A}$ is the closure of $\mathscr{A}_{\mathsf{fp}}$ under filtered colimits.

Example 2.1

(i) Sets, posets and graphs form locally finitely presentable categories, and our assumptions about monomorphisms hold in these categories. Finite presentability of objects means precisely that they are finite.

(ii) If $\mathscr{A}$ is locally finitely presentable, then so is $\mathsf{Fun}_f(\mathscr{A})$, the category of all finitary endofunctors and natural transformations. In case $\mathscr{A} = \mathsf{Set}$, the polynomial endofunctor

$$H_\Sigma X = \coprod_{\sigma\in\Sigma} X^n \qquad n = \text{arity of } \sigma \tag{4}$$

is a finitely presentable object of $\mathsf{Fun}_f(\mathsf{Set})$ iff Σ is a finite set. This is easily seen using Yoneda Lemma. In fact, the finitely presentable objects of $\mathsf{Fun}_f(\mathsf{Set})$ are precisely quotients $H_\Sigma/\!\sim$ of the polynomial functors with Σ finite, where $\sim$ is a congruence of H_Σ, see [5].

Notice that our assumptions concerning monomorphisms carry over to $\mathsf{Fun}_f(\mathscr{A})$ since coproducts are formed objectwise and natural transformations are monic iff their components are monic.

Remark 2.2 We shall need to work with categories that are locally finitely presentable but where the assumptions on monomorphisms above need not hold:

(i) The category
$$\mathsf{Mon}_f(\mathscr{A})$$
of all finitary monads on $\mathscr{A}$ and monad morphisms. This is a locally finitely presentable category. Indeed, as observed by Steve Lack [18], the forgetful functor
$$\mathsf{Mon}_f(\mathscr{A}) \to \mathsf{Fun}_f(\mathscr{A})$$
is finitary and monadic, thus, the local finite presentability of $\mathsf{Fun}_f(\mathscr{A})$ implies that of $\mathsf{Mon}_f(\mathscr{A})$, see [7], 2.78. It follows that filtered colimits of finitary monads are formed object-wise on the level of $\mathscr{A}$.

(ii) We will also make use of the fact that for every locally finitely presentable category $\mathscr{B}$ and object B the coslice category $B/\mathscr{B}$ of all morphisms with domain B is a locally finitely presentable category, see [7], 2.44.

Free monad. Recall from [3] that since H is a finitary endofunctor, free H-algebras $\varphi_X : H(F^H X) \to F^H X$ exist for all objects X of $\mathscr{A}$. Denote by $\widehat{\eta}_X : X \to F^H X$ the universal arrow. As proved by M. Barr [8] the corresponding monad on $\mathscr{A}$
$$F^H$$
of free H-algebras is a free monad on H. It follows that F^H is a finitary monad, and its unit
$$\widehat{\eta} : Id \to F^H$$
together with the natural transformation
$$\varphi : HF^H \to F^H$$
given by the above algebra structures φ_X yield the universal arrow
$$\widehat{\kappa} = (H \xrightarrow{H\widehat{\eta}} HF^H \xrightarrow{\varphi} F^H).$$
The universal property states that for every monad S and every natural transformation $f : H \to S$ there exists a unique monad morphism $\overline{f} : F^H \to S$ such that the triangle below commutes:
$$\begin{array}{ccc} H & \xrightarrow{\widehat{\kappa}} & F^H \\ & \searrow^{f} & \downarrow{\scriptstyle \overline{f}} \\ & & S \end{array} \tag{5}$$

Moreover, from [3] we have

$$F^H = HF^H + Id \quad \text{with injections } \varphi \text{ and } \widehat{\eta}. \tag{6}$$

Remark 2.3 The category $\mathsf{Mon}_f(\mathscr{A})$, being locally finitely presentable, has coproducts. We use the notation $\oplus$ for this coproduct.

Given finitary endofunctor H and K, since the free monad on $H + K$ is the coproduct of the corresponding free monads, we have

$$F^{H+K} = F^H \oplus F^K. \tag{7}$$

We shall use the same notation φ, $\widehat{\eta}$ and $\widehat{\kappa}$ for different endofunctors than H, e. g. $\widehat{\kappa} : H + K \to F^{H+K}$.

Free Completely Iterative Monad. For every object X the functor $H(-)+X$, being finitary, has a terminal coalgebra

$$T^H X \to H\Big(T^H X\Big) + X. \tag{8}$$

By Lambek's lemma [19], this morphism is invertible, and we denote the components of the inverse by

$$\tau_X : H\left(T^H X\right) \to T^H X \qquad \text{and} \qquad \eta_X : X \to T^H X.$$

respectively.

Notation 2.4 *Since $T^H X$ is only used for the given functor H throughout the paper, we omit the upper index H, and write from now on simply*

$$TX.$$

As proved in [1], T is the underlying functor of a monad with the unit $\eta : Id \to T$ above and the multiplication $\mu : TT \to T$. This monad is, moreover, the free completely iterative monad on H, see [1,20]. The above natural transformation $\tau : HT \to T$ yields the universal arrow

$$\kappa = (\, H \xrightarrow{H\eta} HT \xrightarrow{\tau} T \,) \tag{9}$$

Moreover, in analogy to (6) above, we have

$$T = HT + Id \qquad \text{with injections } \tau \text{ and } \eta. \tag{10}$$

Also recall from loc. cit. that the monad multiplication μ is a homomorphism of H-algebras (here we drop objects in the square below as all arrow are

natural transformations):

$$\begin{array}{ccc} HTT & \xrightarrow{\tau T} & TT \\ {\scriptstyle H\mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ HT & \xrightarrow[\tau]{} & T \end{array} \tag{11}$$

Notation 2.5 (i) We denote by

$$H/\mathsf{Mon}_f(\mathscr{A})$$

the category of *H-pointed monads*, i. e., pairs (S, σ) where S is a finitary monad on $\mathscr{A}$ and $\sigma : H \to S$ is a natural transformation. This is a finitely presentable category by Remark 2.2 because it is isomorphic to the coslice category of F^H:

$$H/\mathsf{Mon}_f(\mathscr{A}) \cong F^H/\mathsf{Mon}_f(\mathscr{A}).$$

For example, F^H and T are H-pointed monads (via the universal arrows).

(ii) For every H-pointed monad (S, σ) we write

$$b = [\mu^S \cdot \sigma S, \eta^S] : HS + Id \to S.$$

Lemma 2.6 (Ghani et al [12]) *For every H-pointed monad (S, σ) the endofunctor $HS + Id$ carries a canonical monad structure whose unit is the coproduct injection $\mathsf{inr} : Id \to HS + Id$ and whose multiplication is given by*

$$\begin{array}{c} (HS + Id)(HS + Id) \\ \| \\ HS(HS + Id) + HS + Id \\ \downarrow{\scriptstyle HSb+HS+Id} \\ HSS + HS + Id \\ \downarrow{\scriptstyle [H\mu^S,HS]+Id} \\ HS + Id \end{array} \tag{12}$$

Remark 2.7 For $HS + Id$ we also have an obvious H-pointing

$$\mathsf{inl} \cdot H\eta^S : H \to HS + Id. \tag{13}$$

This defines an endofunctor $\mathcal{H} : H/\mathsf{Mon}_f(\mathscr{A}) \to H/\mathsf{Mon}_f(\mathscr{A})$ on objects by

$$\mathcal{H}(S, \sigma) = (HS + Id, \mathsf{inl} \cdot H\eta^S),$$

see [12] or [21], Lemma 5.2 for details.

Example 2.8 For every finitary endofunctor V we consider F^{H+V} as an H-pointed monad via

$$H \xrightarrow{\mathsf{inl}} H + V \xrightarrow{\widehat{\kappa}} F^{H+V}$$

And $\mathcal{H}(F^{H+V}) = HF^{H+V} + Id$ is then an H-pointed monad via (13) which has the form

$$\psi = (H \xrightarrow{H\widehat{\eta}} HF^{H+V} \xrightarrow{\mathsf{inl}} HF^{H+V} + Id). \tag{14}$$

The proof of the following theorem is similar to the proof of Lemma 2.6 in [12]. The precise statement using the category $H/\mathsf{Mon}_f(\mathscr{A})$ can be found in [21], Theorem 5.4.

Theorem 2.9 *The terminal coalgebra for $\mathcal{H}$ is given by the H-pointed monad T, H-pointed as in (9), and the coalgebra structure $T \xrightarrow{\sim} \mathcal{H}T$ from (8).*

Definition 2.10 A *recursive program scheme* (or *rps* for short) of type H is a natural transformation

$$e : V \to F^{H+V}$$

from an endofunctor V which is a finitely presentable object of $\mathsf{Fun}_f(\mathscr{A})$ to the free monad on $H + V$. It is called *guarded* provided that it factorizes through the summand $HF^{H+V} + Id$ of the coproduct (6):

$$F^{H+V} = (H + V)F^{H+V} + Id = HF^{H+V} + VF^{H+V} + Id,$$

that is, we have a commutative triangle

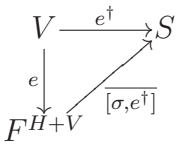
$$\begin{array}{ccc} V & \xrightarrow{\quad e \quad} & F^{H+V} \\ & \searrow^{e_0} & \uparrow{\scriptstyle [\varphi\cdot\mathsf{inl},\widehat{\eta}]} \\ & & HF^{H+V} + Id \end{array} \tag{15}$$

Observe that e_0 is unique since the vertical arrow, being a coproduct injection, is monic. This implies that e_0 and e are in bijective correspondence, which is the reason for our assumptions that $\mathscr{A}$ has monic coproduct injections.

Example 2.11 In case of a polynomial endofunctor $H = H_\Sigma : \mathsf{Set} \to \mathsf{Set}$ every recursive program scheme (3) yields a natural transformation $e : H_\Phi \to F^{H_\Phi + H_\Sigma}$, as explained in the introduction. This is a special case of Definition 2.10: in lieu of a general finitely presentable endofunctor V which is a quotient of H_Σ (cf. Example 2.1(iv)) we just take $V = H_\Sigma$.

The system (3) is guarded iff every right-hand side term is either just a variable or it has an operation symbol from Σ at the head of the term. Such a recursive program scheme is said to be in *Greibach normal form*. All reasonable rps, e. g. (1), are guarded. The unguarded ones such as $f(x) = f(x)$ are to be avoided if we want to work with unique solutions.

Definition 2.12 By a *solution* of a recursive program scheme $e : V \to F^{H+V}$ in an H-pointed monad (S, σ) is meant a natural transformation $e^\dagger : V \to S$ such that the unique monad morphism extending $[\sigma, e^\dagger] : H + V \to S$ makes the triangle below commutative:



(16)

Remark 2.13 (1) Every guarded recursive program scheme (15) turns F^{H+V} into a coalgebra for $\mathcal{H}$. Indeed, $e_0 : V \to \mathcal{H}(F^{H+V})$ together with the pointing ψ, see (14), yield a natural transformation $[\psi, e_0] : H + V \to \mathcal{H}(F^{H+V})$ which, by the universal property of the free monad F^{H+V}, provides a unique monad morphism

$$\overline{[\psi, e_0]} : F^{H+V} \to \mathcal{H}(F^{H+V}) \tag{17}$$

It preserves the pointing: we have

$$\overline{[\psi, e_0]} \cdot (\widehat{\kappa} \cdot \mathsf{inl}\,) = [\psi, e_0] \cdot \mathsf{inl} = \psi.$$

Thus, F^{H+V} is a coalgebra.

(2) Conversely, every coalgebra for $\mathcal{H}$ carried by F^{H+V}, where V is a finitely presentable endofunctor, stems from a guarded recursive program scheme: the coalgebra structure $r : F^{H+V} \to \mathcal{H}(F^{H+V})$ is uniquely determined by $r \cdot \widehat{\kappa} : H + V \to \mathcal{H}(F^{H+V})$, and the left-hand component of $r \cdot \widehat{\kappa}$ being the pointing ψ, we see that r is determined by $e_0 = r \cdot \widehat{\kappa} \cdot \mathsf{inr} : V \to \mathcal{H}(F^{H+V})$ defining a (unique) recursive program scheme.

(3) For the terminal coalgebra T for $\mathcal{H}$, see Theorem 2.9, we thus obtain the unique coalgebra homomorphism

$$e^* : F^{H+V} \to T. \tag{18}$$

Remark 2.14 Our concept of a recursive program scheme is a special case of the algebraic systems studied by Neil Ghani et al [11]. Let us recall from that paper that

(i) an H-pointed monad is called *coalgebraic* if it is isomorphic to the monad $HS + Id$ of Lemma 2.6 via $b : HS + Id \to S$ in Notation 2.5(ii),

(ii) examples of coalgebraic monads are F^H, see (6), and T, see (10),

(iii) T is the final coalgebraic monad; we denote by $u_S : S \to T$ the unique morphism for a coalgebraic monad (S, σ),

(iv) an *algebraic system* is given by a finitary monad E, a coalgebraic monad

(S, σ) and a monad morphism

$$e : E \to H(S \oplus E) + Id,$$

(v) a *solution* of e is a monad morphism $s : E \to T$ such that the square below commutes:

$$\begin{array}{ccc} E & \xrightarrow{\quad s \quad} & T \\ {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle [\tau,\eta]^{-1}} \\ H(S \oplus E) + Id & \xrightarrow[H([u_S,s])+Id]{} & HT + Id \end{array}$$

Theorem 2.15 (Ghani et al [11]) *Every algebraic system has a unique solution.*

This gives a solution theorem for recursive program schemes as follows: due to (7) we have the morhism $e_0 : V \to H(F^H \oplus F^V) + Id$ in (15) yielding an algebraic system via (5):

$$\overline{e_0} : F^V \to H(F^H \oplus F^V) + Id. \tag{19}$$

Indeed, take $E = F^V$ and $S = F^H$. Thus, a unique solution $s : F^V \to T$ exists.

Theorem 2.16 *Every guarded recursive program scheme of type H has a unique solution $e^\dagger$ in T. It can be computed from the unique coalgebra homomorphism $e^* : F^{H+V} \to T$ by*

$$e^\dagger = (V \xrightarrow{\mathsf{inr}} H + V \xrightarrow{\widehat{\kappa}} F^{H+V} \xrightarrow{e^*} T). \tag{20}$$

Indeed, for the unique solution $s : F^V \to T$ of the algebraic system $\overline{e_0}$ in (19) above we obtain a solution $e^\dagger$ in the sense of Definitinon (2.10) by composing with $\widehat{\kappa} : V \to F^V$:

$$e^\dagger = (\, V \xrightarrow{\widehat{\kappa}} F^V \xrightarrow{s} T\,).$$

The proof that (16) commutes is performed using some diagram chasing. A somewhat subtle point is that for $u_S : S \to T$ (see Remark 2.14(iii)) we have the equality

$$[u_S, s] = \overline{[\widehat{\kappa}, e^\dagger]} : F^{H+V} \to T.$$

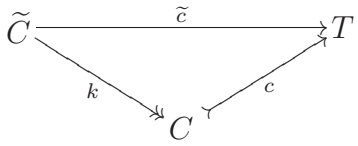
Here the square brackets on the left refer to $\oplus$ in $H/\mathsf{Mon}_f(\mathscr{A})$ and those on the right to $+$ in $\mathsf{Fun}_f(\mathscr{A})$. The verification uses the universal property of the free monad on $H + V$ and is not difficult. The fact that (20) holds follows from the same diagram.

The uniqueness is also a consequence of the fact that for any solution $e^\dagger$ in the sense of Definition 2.10 its extension $\overline{e^\dagger} : F^V \cdot T$ is a solution of the corresponding algebraic system $\overline{e_0}$.

Remark 2.17 It is our goal to define a finitary submonad C of T formed by all solutions of recursive program schemes of type H. We do this in two steps.

(i) A finitary monad $\widetilde{C}$ together with a monad morphism $\widetilde{c} : \widetilde{C} \to T$ is constructed by forming a colimit of coalgebras for the endofunctor $\mathcal{H}$ obtained from all recursion program schemes.

(ii) The (strong epi, mono)-factorization (cf. Lemma 2.19 below) of $\widetilde{c}$ is formed to obtain the desired submonad:



This factorization is going to be performed in the category

$$\mathsf{Mon}_c(\mathscr{A})$$

of all monads on $\mathscr{A}$ that are *countably accessible*, that is, the underlying functors preserve countably filtered colimits. (Recall that a countably filtered category is such that every subcategory with countably many objects and morphisms has a cocone in it.) The monad $\widetilde{C}$ lies in $\mathsf{Mon}_c(\mathscr{A})$ because it is finitary by construction, and we use the following

Lemma 2.18 *For every finitary endofunctor H the monad T (see Notation 2.4) is countably accessible.*

Proof. It is proved in Proposition 5.16 of [4] that TZ can be constructed as the colimit of the diagram of all coalgebras for $H(-) + Z$ carried by all countably presentable objects. Thus, T coincides with the countably accessible monad R^λ of loc. cit. for the first uncountable ordinal λ. □

Lemma 2.19 *In the categories $\mathsf{Mon}_f(\mathscr{A})$ and $\mathsf{Mon}_c(\mathscr{A})$ every morphism has a (strong epi, mono)-factorization, and monomorphisms are precisely the monad morphisms whose components are monic in $\mathscr{A}$.*

Proof. The first statement follows from the fact that both $\mathsf{Mon}_f(\mathscr{A})$ and $\mathsf{Mon}_c(\mathscr{A})$ are locally presentable categories. See [18] for $\mathsf{Mon}_f(\mathscr{A})$, the argument for $\mathsf{Mon}_c(\mathscr{A})$ is analogous. Every locally presentable category has (strong epi, mono)-factorizations by [7], 1.61.

The second statement follows, in case of $\mathsf{Mon}_f(\mathscr{A})$, from the fact that in the category $\mathsf{Fun}_f(\mathscr{A}) \cong [\mathscr{A}_{fp}, \mathscr{A}]$ monomorphisms are precisely the morphisms with components monic in $\mathscr{A}$, and the forgetful functor $\mathsf{Mon}_f(\mathscr{A}) \to \mathsf{Fun}_f(\mathscr{A})$

has the left adjoint $H \mapsto F^H$. Analogously for the category $\mathsf{Fun}_c(\mathscr{A})$ of countably accessible endofunctors: this is equivalent to $[\mathscr{A}_c, \mathscr{A}]$ where $\mathscr{A}_c$ is a full subcategory representing all countably presentable objects in $\mathscr{A}$ (i. e., such that the hom-functor is countably accessible). And once again, every countably accessible functor H generates a countably accessible free monad F, see [3], yielding a left adjoint of the forgetful functor $\mathsf{Mon}_c(\mathscr{A}) \to \mathsf{Fun}_c(\mathscr{A})$.□

Corollary 2.20 *The category $H/\mathsf{Mon}_f(\mathscr{A})$ has (strong epi, mono)-factorizations, and the functor $\mathcal{H}$ preserves monomorphisms.*

Indeed, the forgetful functor $H/\mathsf{Mon}_f(\mathscr{A}) \to \mathsf{Mon}_f(\mathscr{A})$ clearly creates (strong epi, mono)-factorizations. Given a monomorphism $m : (S, \sigma) \to (S', \sigma')$ in $H/\mathsf{Mon}_f(\mathscr{A})$, then m is componentwise monic, thus, so is Hm (since H preserves monomorphisms), and so is also $\mathcal{H}m = Hm + id$ (since coproducts of monomorphisms are monic in $\mathscr{A}$).

Construction 2.21 The H-pointed monad $\widetilde{C}$. For every guarded recursive program scheme (15) consider F^{H+V} as a coalgebra for the functor $\mathcal{H}$, see (17).

We denote by

$$\mathsf{EQ}_0 \subseteq \mathsf{Coalg}\,\mathcal{H}$$

the full subcategory of all these coalgebras. It is essentially small since $\mathsf{Fun}_f(\mathscr{A})$ has only a set of finitely presentable objects up to isomorphism. We denote the colimit of this small diagram by

$$\widetilde{C}^H = \operatorname{colim} \mathsf{EQ}_0 \qquad (\text{in } \mathsf{Coalg}\,\mathcal{H}).$$

Thus, we have a finitary monad $\widetilde{C}$ with an H-pointing and a coalgebra structure denoted by

$$\rho : H \to \widetilde{C}^H \qquad \text{and} \qquad r : \widetilde{C}^H \to \mathcal{H}(\widetilde{C}^H)$$

respectively, together with a colimit cocone

$$e^\sharp : F^{H+V} \to \widetilde{C}^H \qquad \text{for all rps } e : V \to F^{H+V},$$

formed by coalgebra homomorphisms for $\mathcal{H}$ preserving the pointing (14), i. e. with

$$\rho = e^\sharp \cdot (\widehat{\kappa} \cdot \mathsf{inl}\,) \qquad \text{for every } e.$$

We see in the next lemma that EQ_0 is a connected category. Since the forgetful functors

$$\mathsf{Coalg}\,\mathcal{H} \to H/\mathsf{Mon}_f(\mathscr{A}) \to \mathsf{Mon}_f(\mathscr{A})$$

clearly preserve connected colimits, the above cocone $e^\sharp : F^{H+V} \to T$ is also a colimit cocone in $\mathsf{Mon}_f(\mathscr{A})$.

Lemma 2.22 EQ_0 *is closed under finite coproducts in* $\mathsf{Coalg}\,\mathcal{H}$.

Proof. Consider two objects of EQ_0 determined by

$$e : V \to HF^{H+V} + Id \qquad \text{and} \qquad e' : V' \to HF^{H+V'} + Id$$

The coproduct injections $i : H + V \to H + V + V'$ and $i' : H + V' \to H + V + V'$ yield corresponding monad morphisms $\widetilde{i} : F^{H+V} \to F^{H+V+V'}$ and $\widetilde{i'} : F^{H+V'} \to F^{H+V}$. Denote by

$$k = ((HF^{H+V} + Id) + (HF^{H+V'} + Id) \xrightarrow{[H\widetilde{i}+Id, H\widetilde{i'}+Id]} HF^{H+V+V'} + Id)$$

the canonical morphism. We prove that the object $f : V + V' \to F^{H+V+V'}$ of EQ_0 determined by

$$f_0 = k \cdot (e_0 + e'_0) : V + V' \to HF^{H+V+V'} + Id$$

is the coproduct of the two given objects.

We know from Remark 2.13 that morphisms from the above object into an $\mathcal{H}$-coalgebra $X = ((S, s), p)$ are given by natural transformations

$$r : V + V' \to S$$

such that the extension $\overline{[s, r]} : F^{H+V+V'} \to S$ of the transformation $[s, r] : H + V + V' \to S$ to a monad morphism fulfils

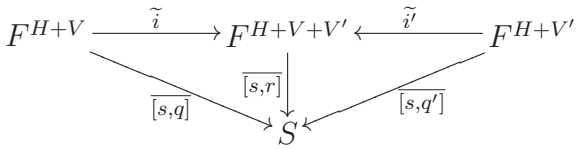
$$p \cdot r = (H\overline{[s, r]} + Id) \cdot f.$$

We claim that this holds for $r : V + V' \to S$ iff

(i) the left-hand component $q : V \to S$ of r gives rise to a morphism of $\mathsf{Coalg}\,\mathcal{H}$ from the object determined by e_0 into X

(ii) and the right-hand component $q' : V' \to S$ yields a morphism from the object determined by e'_0 into X.

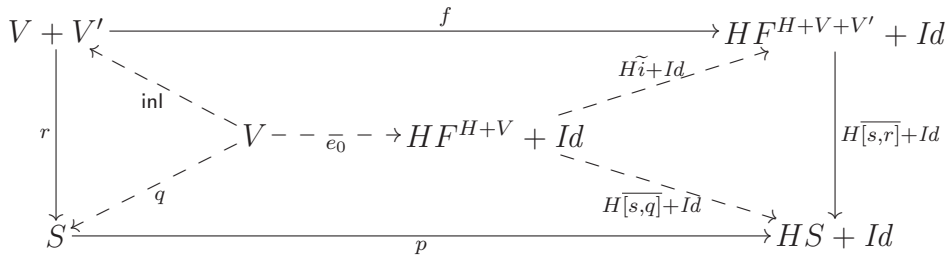
For that observe first that the diagram



commutes: indeed, all these morphisms are monad morphisms. The left-hand triangle commutes since $\widetilde{i} \cdot \widehat{\kappa}^{H+V} = \widehat{\kappa}^{H+V+V'} \cdot i$, therefore,

$$(\overline{[s, r]} \cdot \widetilde{i}) \cdot \widehat{\kappa} = [s, r] \cdot i = [s, q] = \overline{[s, q]} \cdot \widehat{\kappa}$$

and analogously for the right-hand triangle. Thus, the square



commutes iff $\overline{[s,q]}$ and $\overline{[s,q']}$ are morphisms of $\mathsf{Coalg}\,\mathcal{H}$ into X: in the diagram we indicated the left-hand component (commuting iff $p \cdot q = (H\overline{[s,q]} + Id) \cdot e_0$, that is, $\overline{q}$ is a homomorphism), analogously for the right-hand one. □

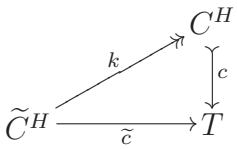
Corollary 2.23 *$\widetilde{C}$ is a filtered colimit of the closure EQ of EQ_0 under coequalizers in $\mathsf{Coalg}\,\mathcal{H}$.*

Indeed, since EQ_0 is closed under finite coproducts, EQ is closed under finite colimits, thus, it is filtered. And $\operatorname{colim}\mathsf{EQ} \cong \operatorname{colim}\mathsf{EQ}_0$.

Definition 2.24 The context-free monad C. Denote by

$$\widetilde{c} : \widetilde{C}^H \to T$$

the unique coalgebra homomorphism and define the *context-free monad* of H as the submonad C^H of T obtained by the following (strong epi, mono)-factorization of $\widetilde{c}$ in $\mathsf{Mon}_c(\mathscr{A})$:



Remark 2.25 (i) Recall from Theorem 2.9 that $\widetilde{c}$ is uniquely defined. Since $\widetilde{C}^H$ is finitary and T countably accessible, see Lemma 2.18, we have the desired factorization by Lemma 2.19.

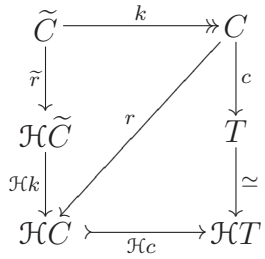
(ii) The context-free monad is pointed: The pointing $\widetilde{\rho} : H \to \widetilde{C}^H$ of $\widetilde{C}^H$ yields the pointing

$$\rho = k \cdot \widetilde{\rho} : H \to C^H$$

of C^H which c preserves (because $\widetilde{c}$ is a morphism of $H/\mathsf{Mon}_f(\mathscr{A})$). Analogously to T we shall write C and $\widetilde{C}$ without the upper index H from now on.

Observation 2.26 The functor $\mathcal{H}$ preserves monomorphisms by Corollary 2.20, thus, C carries a canonical structure r of an $\mathcal{H}$-coalgebra derived from

the structure $\widetilde{r}$ for $\widetilde{C}$:



(21)

Indeed, recall that $c \cdot k = \widetilde{c}$ is an $\mathcal{H}$-coalgebra homomorphism; so the outside of the above square commutes, and we can use the unique diagonalization property of the factorization system to obtain r.

Theorem 2.27 *Every guarded recursive program scheme $e : V \to F^{H+V}$ has a unique solution in the context-free monad of H.*

Proof. We use $e^\ddagger$ for solutions in C and $e^\dagger$ for solutions in T throughout this proof. We are to prove that there exists a unique natural transformation $e^\ddagger : V \to C$ with $e^\ddagger = \overline{[\rho, e^\ddagger]} \cdot e$. Recall that the colimit injection $e^\sharp : F^{H+V} \to \widetilde{C}$ in Construction 2.21 is a coalgebra homomorphism for $\mathcal{H}$, hence, so is $\widetilde{c} \cdot e^\sharp$, which proves

$$e^* = \widetilde{c} \cdot e^\sharp,$$

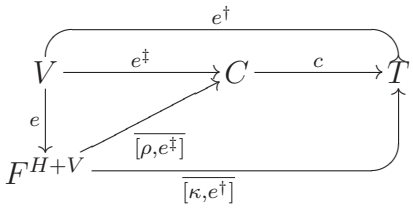
see Theorem 2.16 (because T is a terminal coalgebra by Theorem 2.9). Therefore, by (20) we have

$$e^\dagger = \widetilde{c} \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr} \; = c \cdot k \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr}\,.$$

Thus for $e^\ddagger = k \cdot e^\sharp \cdot \widehat{\kappa} \cdot \mathsf{inr}$ we obtain

$$e^\dagger = c \cdot e^\ddagger.$$

We conclude that $e^\ddagger$ is the desired solution in C: in the following diagram



the outside commutes, see (16) with $\sigma = \kappa$, and the right-hand part does since $\kappa = c \cdot \rho$ (see Definition 2.24). Consequently, the left-hand triangle commutes: recall from Definition 2.24 that c is a monomorphism.

The uniqueness follows from the same diagram: if the left-hand triangle commutes, so does the outside, and since $e^\dagger$ is uniquely determined (see Theorem 2.16), we conclude $e^\dagger = c \cdot e^\ddagger$. Finally, use again that c is monic. □

3 The context-free monad is ideal

Under the assumptions of Section 2 we prove that C is an ideal monad in the sense of C. Elgot [10] for every finitary endofunctor H. Elgot's concept was defined for monads (S, η, μ) in Set: the monad is ideal if the complement of $\eta : Id \to S$ is a subfunctor $\sigma : S' \hookrightarrow S$ of S (thus, $S = S' + Id$) and μ restricts to a natural transformation $\mu' : S'S \to S'$. For general categories "ideal" is not a property but a structure:

Definition 3.1 ([1]) An *ideal monad* is a sixtuple $(S, \eta, \mu, S', \sigma, \mu')$ where (S, η, μ) is a monad,

$$\sigma : S' \to S \qquad (\text{"the ideal"})$$

is a subfunctor such that $S = S' + Id$ with injection σ and η, and

$$\mu' : S'S \to S'$$

is a natural transformation restricting μ in the sense that

$$\mu \cdot \sigma S = \sigma \cdot \mu'$$

Example 3.2

(i) The free monad F^H is ideal: its ideal is HF^H, see (6).

(ii) The free completely iterative monad T is ideal: its ideal is HT, see (10).

Remark 3.3 It is our goal to prove that the context-free monad (C, η^C, μ^C) is ideal. The $\mathcal{H}$-coalgebra structure $r : C \to HC + Id$, see Observation 2.26, is (analogously to the two examples F^H and T above) invertible, as we prove below: its inverse is the morphism

$$b \equiv HC + Id \xrightarrow{\rho C + Id} CC + Id \xrightarrow{[\mu^C, \eta^C]} C \tag{22}$$

From that we will derive that C is an ideal monad with the ideal

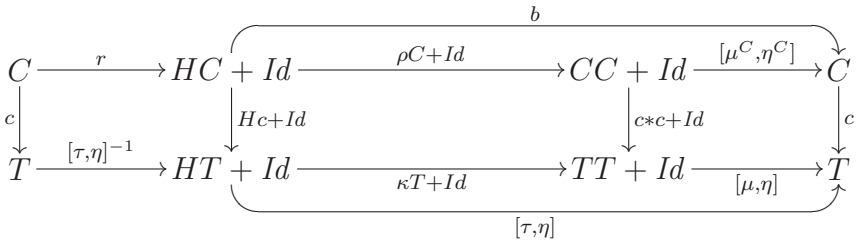
$$b \cdot \mathsf{inl} : HC \to C$$

Theorem 3.4 *The context-free monad C is an ideal monad for every H.*

Proof. We first prove $r = b^{-1}$.

(1) The proof of $b \cdot r = id$ follows, since c is a monomorphism, from the commutativity of the following diagram (here $c * c$ denotes the parallel

composition of natural transformations):



Indeed, the right-hand square commutes since $c : C \to T$ is a monad morphism, the left-hand one does because c is a coalgebra homomorphism for $\mathcal{H}$ (see (21)), and the middle square follows from fact that by Remark 2.25 c preserves the pointing, i.e., $c \cdot \rho = \tau \cdot H\eta$. Finally, the lower part follows from (11):

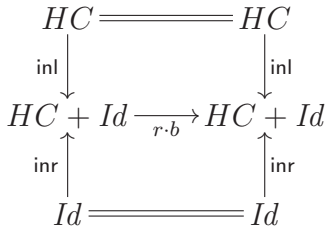
$$\mu \cdot \tau T \cdot H\eta T = \tau \cdot H\mu \cdot H\eta T = \tau.$$

So the outside of the diagram commutes:

$$c \cdot b \cdot r = c,$$

and since c is a monomorphism, we see that $b \cdot r = id$.

(2) To prove that $r \cdot b = id$ we show that the diagram below commutes:

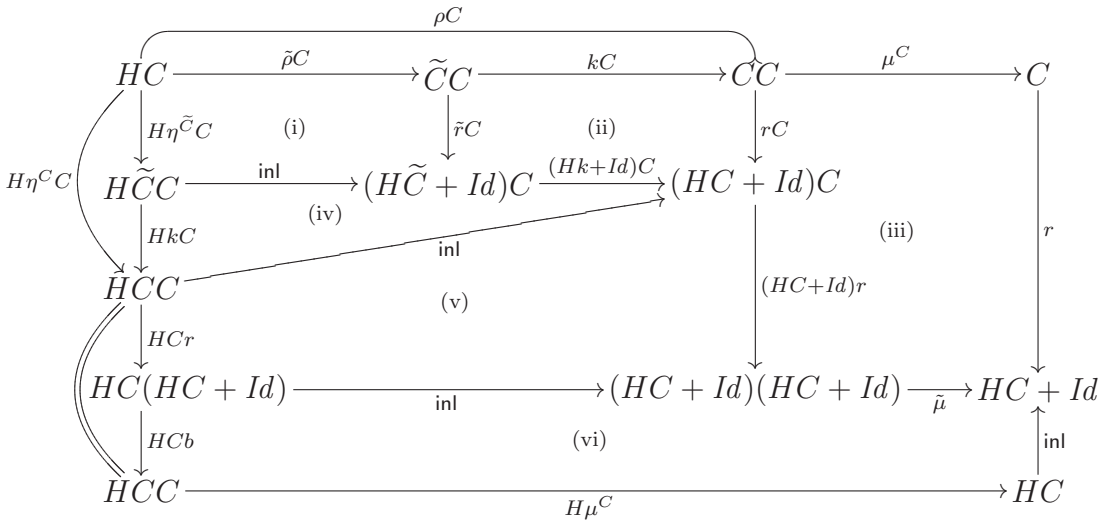


For the commutativity of the lower square we have since r is a monad morphism and the unit of the monad in the codomain is, by Lemma 2.6, inr that

$$r \cdot b \cdot \mathsf{inr} = r \cdot \eta^C = \mathsf{inr}\,.$$

Since $b \cdot \mathsf{inl} = \mu^C \cdot \rho C = \mu^C \cdot (kC \cdot \widetilde{\rho} C)$, the commutativity of the upper square

boils down to showing that the outside of the following diagram commutes:

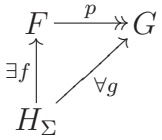


Here $\widetilde{\mu}$ denotes the monad multiplication (12) of Lemma 2.6, where $S = C$ and $\sigma = \rho$. Indeed, all inner parts commute: the two left-hand parts commute since $k \cdot \eta^{\widetilde{C}} = \eta^C$ and $b \cdot r = id$, for part (i) recall that the coalgebra structure $\tilde{\rho}$ is a morphism in $H/\mathsf{Mon}_f(\mathscr{A})$, part (ii) commutes since k is a coalgebra homomorphism for $\mathcal{H}$, for (iii) use that r is a monad morphism, (iv) and (v) are trivial, and part (vi) commutes by (12). The remaining upper part commutes since k preserves the H-pointing. Finally, using the monad law $\mu^C \cdot \eta^C C = id$, we get $r \cdot \mu^C \cdot \rho C = \mathsf{inl} : HC \to HC + Id$, and this completes the proof. □

4 Context-free trees

We now return to the original concept of a context-free (or algebraic) Σ-tree on a given signature Σ, as studied by Bruno Courcelle, see the introduction. We prove that the context-free monad C^{H_Σ} of the polynomial endofunctor H_Σ of Set is indeed precisely the submonad $C^{H_\Sigma} \hookrightarrow T^{H_\Sigma}$ of the Σ-tree monad consisting of all context-free Σ-trees of Definition 1.1.

Observation 4.1 Polynomial endofunctors are projective in $\mathsf{Fun}_f(\mathsf{Set})$. That is, for every epimorphism (which means a componentwise surjective natural transformation) $p : F \to G$ and every natural transformation $g : H_\Sigma \to G$ there exists a natural transformation $f : H_\Sigma \to F$ with $g = p \cdot f$:



In case Σ consists of a single n-ary symbol, this follows from Yoneda Lemma, since $H_\Sigma \cong \mathsf{Set}(n, -)$: the natural transformation g corresponds to an element of Gn, and we find its inverse image (under p_n) in Fn, giving us $f : H_\Sigma \to F$. If Σ has more symbols, apply Yoneda Lemma to each of them separately.

Theorem 4.2 *For every signature Σ we have:*

$$C^{H_\Sigma} = \textit{the monad of context-free } \Sigma\textit{-trees}$$

Proof. Throughout the proof we write H in lieu of H_Σ and C in lieu of C^{H_Σ}.

(1) We prove that every element of CX lies in the image of $e^\ddagger$ for some guarded recursive program scheme

$$e : H_\Phi \to F^{H+H_\Phi}$$

where $e^\ddagger$ is the unique solution in C, see Theorem 2.27.

Indeed, since $\widetilde{C}$ is the filtered colimit of EQ, see Corollary 2.23, and filtered colimits in $\mathsf{Mon}_f(\mathscr{A})$ (and thus also in $H/\mathsf{Mon}_f(\mathscr{A})$) are computed on the level of the underlying functors (in other words: filtered colimits are formed object-wise in $\mathscr{A}$), we have for every set X a colimit cocone

$$r^\sharp_X : SX \to \widetilde{C}X$$

where $s : (S, \sigma) \to \mathcal{H}(S, \sigma)$ ranges over all coalgebras in EQ and $s^\sharp : S \to \widetilde{C}$ is the colimit cocone.

Since EQ is a closure of EQ_0 under coequalizers, every object of EQ is a quotient of one in EQ_0. Thus, we have a guarded recursive program scheme

$$e : V \to F^{H+V} \tag{23}$$

and an epimorphic coalgebra homomorphism for $\mathcal{H}$:

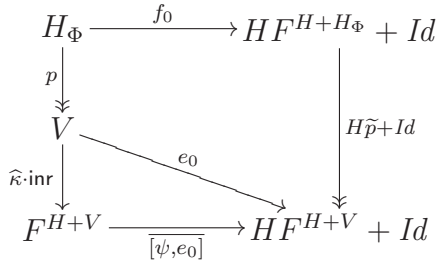
$$\begin{array}{ccc} (F^{H+V}, \widehat{\kappa} \cdot \mathsf{inl}\,) & \longrightarrow & \mathcal{H}(F^{H+V}, \widehat{\kappa} \cdot \mathsf{inl}\,) \\ \downarrow{\scriptstyle q} & & \downarrow{\scriptstyle \mathcal{H}q} \\ (S, \sigma) & \xrightarrow{\quad s \quad} & \mathcal{H}(S, \sigma) \end{array}$$

Since V is a finitely presentable functor, there exists by Example 2.1(ii) a finite signature Φ and an epimorphic natural transformation

$$p : H_\Phi \to V.$$

The free-monad functor takes $H + p : H + H_\Phi \to H + V$ to a monad morphism $\widetilde{p} : F^{H+H_\Phi} \to F^{H+V}$ which is also an epimorphism (since the free-monad

functor is a left adjoint). Due to the projectivity of H_Φ we obtain a natural transformation f_0 making the diagram

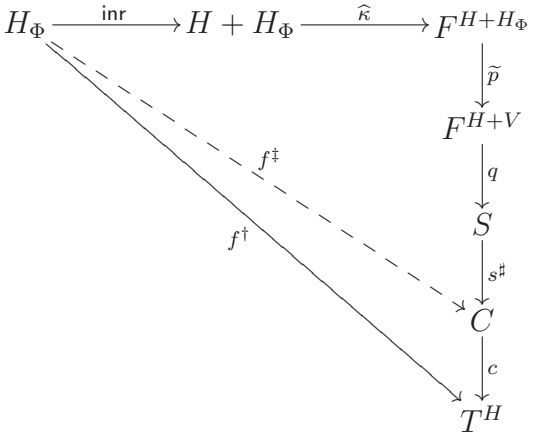


commutative (see Observation 4.1.) Here f_0 is the guard of a "classical" guarded recursive program scheme

$$f : H_\Phi \to F^{H+H_\Phi}$$

and for the corresponding $\mathcal{H}$-coalgebra on F^{H+H_Φ}, see Remark 2.13, the above monad morphism $\widetilde{p}$ is a coalgebra homomorphism.

We conclude that the triangles for $f^\dagger$ (see Theorem 2.16) and $f^\ddagger$ (see Theorem 2.27)



commute: recall from (20) that the coalgebra homomorphism f^* fulfils

$$f^\dagger = f^* \cdot \widehat{\kappa} \cdot \mathsf{inr}\,,$$

and so we only need to notice that the vertical arrow, being a coalgebra homomorphism, is equal to f^*. Since c is a monomorphism, the upper triangle also commutes. Thus, every element in the image of $s^\sharp_X$ lies in the image of $f^\ddagger_X$ for the above recursive program scheme f.

(2) We will verify that $c_X : CX \hookrightarrow TX$ consists precisely of the context-free Σ-trees on X. Indeed, every context-free Σ-tree has the form

$$t = e^\dagger_X(x)$$

for some guarded recursive program scheme $e : H_\Phi \to F^{H+H_\Phi}$ and since $e^\dagger_X = c_X \cdot e^\ddagger_X$, the tree t lies in CX.

Conversely every element of CX has, by item (1) above, the form $e^\ddagger_X(x)$ for some guarded rps $e : H_\Phi \to F^{H+H_\Phi}$. □

5 Conclusions and Open Problems

The aim of our paper was to construct for a finitary endofunctor H a monad expressing solutions of recursive program schemes of type H. We hoped originally to achieve what we managed to do for the first-order recursive equations of type H in previous work [4]: there we defined the rational monad R^H based on solutions of recursive equations, we proved that R^H is iterative (and, in particular, ideal) in the sense of Calvin Elgot, and we characterized R^H as the free iterative monad on H. From this we derived, in case of endofunctors of Set, that R^H is closed under second-order substitution. Moreover, the construction worked for all locally finitely presentable base categories.

In the present paper we also exhibited a general construction: for every finitary endofunctor H we provided a context-free monad C^H based on solutions of recursive program schemes of type H. The existence and uniqueness of these solutions were derived from the corresponding more general solution theorem of Ghani et al [11]. In case H is actually a polynomial endofunctor of Set associated to a signature Σ, our monad coincides with the monad of context-free (= algebraic) trees of Bruno Courcelle [9]. However, whereas Courcelle proved that the context-free-tree monad is iterative, we were only able to prove that the general context-free monad is ideal.

In fact, as soon as C^H would be proved to be iterative, the intuition says that this is not enough: the next open problem is, then, whether C^H is closed under second-order substitution in the sense of [21]. Again, this was, for context-free Σ-trees, proved by Bruno Courcelle.

Finally, the rational monad R^H and the monad T^H are both characterized by universal properties; R^H is the free iterative monad and T^H the free completely iterative one. It remains to be seen whether C^H can be characterized by some universal property, too. Unfortunately, context-free trees cannot serve as a guiding example in this respect as no universal property of them is known.

References

[1] P. Aczel, J. Adámek, S. Milius, and J. Velebil. Infinite trees and completely iterative theories: A coalgebraic view. *Theoret. Comput. Sci.*, 300:1–45, 2003.

[2] P. Aczel, J. Adámek, and J. Velebil. A coalgebraic view of infinite trees and iteration. In *Proc. Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44 of *Electron. Notes Theor. Comput. Sci.*, pages 1–26, 2001.

[3] J. Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolin.*, 15:589–602, 1974.

[4] J. Adámek, S. Milius, and J. Velebil. Iterative algebras at work. *Math. Structures Comput. Sci.*, 16(6):1085–1131, 2006.

[5] J. Adámek, S. Milius, and J. Velebil. Semantics of higher-order recursion schemes. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *Proc. Coalgebraic and Algebraic Methods in Computer Science (CALCO'09)*, volume 5728 of *Lecture Notes Comput. Sci.*, pages 49–63. Springer, 2009.

[6] J. Adámek, S. Milius, and J. Velebil. Iterative reflections of monads. to appear in *Math. Structures in Comput. Sci.*, published online by Cambridge University Press, doi:10.1017/S0960129509990326, February 2010.

[7] J. Adámek and J. Rosický. *Locally presentable and accessible categories*. Cambridge University Press, 1994.

[8] M. Barr. Coequalizers and free triples. *Math. Z.*, 116:307–322, 1970.

[9] B. Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25:95–169, 1983.

[10] C. C. Elgot. Monadic computation and iterative algebraic theories. In H. E. Rose and J. C. Sheperdson, editors, *Logic Colloquium '73*, Amsterdam, 1975. North-Holland Publishers.

[11] N. Ghani, C. Lüth, and F. D. Marchi. Solving algebraic equations using coalgebra. *Theor. Inform. Appl.*, 37:301–314, 2003.

[12] N. Ghani, C. Lüth, and F. D. Marchi. Monads of coalgebras: rational terms and term graphs. *Math. Structures Comput. Sci.*, 15(3):433–451, 2005.

[13] N. Ghani, C. Lüth, F. D. Marchi, and A. J. Power. Algebras, coalgebras, monads and comonads. In *Proc. Coalgebraic Methods in Computer Science (CMCS'01)*, volume 44 of *Electron. Notes Theor. Comput. Sci.*, pages 128–145, 2001.

[14] N. Ghani, C. Lüth, F. D. Marchi, and A. J. Power. Dualizing initial algebras. *Math. Structures Comput. Sci.*, 13(2):349–370, 2003.

[15] N. Ghani and T. Uustalu. Coproducts of ideal monads. *Theor. Inform. Appl.*, 38(4):321–342, 2004.

[16] S. Ginali. Regular trees and the free iterative theory. *J. Comput. System Sci.*, 18:228–242, 1979.

[17] I. Guessarian. *Algebraic Semantics*, volume 99 of *Lecture Notes in Comput. Sci.* Springer, 1981.

[18] S. Lack. On the monadicity of finitary monads. *J. Pure Appl. Algebra*, 140:65–73, 1999.

[19] J. Lambek. A fixpoint theorem for complete categories. *Math. Z.*, 103:151–161, 1968.

[20] S. Milius. Completely iterative algebras and completely iterative monads. *Inform. and Comput.*, 196:1–41, 2005.

[21] S. Milius and L. S. Moss. The category theoretic solution of recursive program schemes. *Theoret. Comput. Sci.*, 366:3–59, 2006.

[22] L. S. Moss. Parametric corecursion. *Theoret. Comput. Sci.*, 260(1–2):139–163, 2001.

Categorifying Computations into Components via Arrows as Profunctors

Kazuyuki Asada Ichiro Hasuo

Research Institute for Mathematical Sciences, Kyoto University, Japan
PRESTO Research Promotion Program, Japan Science and Technology Agency
http://www.kurims.kyoto-u.ac.jp/~{asada,ichiro}

Abstract

The notion of *arrow* by Hughes is an axiomatization of the algebraic structure possessed by structured computations in general. We claim that an arrow also serves as a basic *component calculus* for composing state-based systems as components—in fact, it is a *categorified* version of arrow that does so. In this paper, following the second author's previous work with Heunen, Jacobs and Sokolova, we prove that a certain coalgebraic modeling of components—which generalizes Barbosa's—indeed carries such arrow structure. Our coalgebraic modeling of components is parametrized by an arrow A that specifies computational structure exhibited by components; it turns out that it is this arrow structure of A that is lifted and realizes the (categorified) arrow structure on components. The lifting is described using the first author's recent characterization of an arrow as an internal strong monad in **Prof**, the bicategory of small categories and *profunctors*.

Keywords: algebra, arrow, coalgebra, component, computation, profunctor

1 Introduction

1.1 Arrow for Computation

In functional programming, the word *computation* often refers to a procedure which is not necessarily *purely functional*, typically involving some *side-effect* such as I/O, global state, non-termination and non-determinism. The most common way to organize such computations is by means of a *(strong) monad* [21], as is standard in Haskell. However side-effect—that is "structured output"—is not the only cause for the failure of pure functionality. A *comonad* can be used to encapsulate "structured input" [26]; the combination of a monad and a comonad via a distributive law can be used for input and output that are both structured. There are much more additional struc-

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

ture that a functional programmer would like to think of as "computations"; Hughes' notion of *arrow* [13] is a general axiomatization of such. [1]

Let $\mathbb{C}$ be a Cartesian category of types and pure functions, in a functional programming sense. The notion of arrow over $\mathbb{C}$ is an algebraic one: it axiomatizes those operators which the set of computations should be equipped with, and those equations which those operators should satisfy. More specifically, an arrow A is

- carried by a family of sets $A(J,K)$ for each $J,K \in \mathbb{C}$;
- equipped with the following three families of operators arr, $\ggg$ and first:

$$\begin{array}{rcll} & \mathsf{arr} f \in A(J,K) & \text{for each morphism } f : J \to K \text{ in } \mathbb{C}, \\ A(J,K) \times A(K,L) & \overset{\ggg_{J,K,L}}{\longrightarrow} & A(J,L) & \text{for each } J,K,L \in \mathbb{C}, \\ A(J,K) & \overset{\mathsf{first}_{J,K,L}}{\longrightarrow} & A(J \times L, K \times L) & \text{for each } J,K,L \in \mathbb{C}; \end{array}$$

- that are subject to several equational axioms: among them is

$$(a \ggg_{J,K,L} b) \ggg_{J,L,M} c = a \ggg_{J,K,M} (b \ggg_{K,L,M} c) \quad \text{for each } a \in A(J,K), b \in A(K,L), c \in A(L,M). \qquad (\ggg\text{-Assoc})$$

The other axioms are presented later in Def. 3.1.

The intuitions are clear: presenting an A-computation from J to K by a box $\overset{J}{\to}\boxed{\ }\overset{K}{\to}$, the three operators ensure that we can combine computations in the following ways.

- (Embedding of pure functions) $\overset{J}{\to}\boxed{\mathsf{arr}\, f}\overset{K}{\to}$
- (Sequential composition) $\left(\overset{J}{\to}\boxed{a}\overset{K}{\to}\ ,\ \overset{K}{\to}\boxed{b}\overset{L}{\to}\right) \overset{\ggg_{J,K,L}}{\longmapsto} \overset{J}{\to}\boxed{a}\overset{K}{\to}\boxed{b}\overset{L}{\to}$
- (Sideline) $\overset{J}{\to}\boxed{a}\overset{K}{\to} \overset{\mathsf{first}_{J,K,L}}{\longmapsto} \left[\begin{array}{c} \overset{J}{\to}\boxed{a}\overset{K}{\to} \\ \overset{L}{\to}\quad\overset{L}{\to} \end{array}\right]$

The ($\ggg$-Assoc) axiom above, for example, ensures that the following compositions of three consecutive A-computations are identical.

$$\overset{J}{\to}\boxed{\boxed{a}\overset{K}{\to}\boxed{b}}\overset{L}{\to}\boxed{c}\overset{M}{\to} \quad = \quad \overset{J}{\to}\boxed{a}\overset{K}{\to}\boxed{\boxed{b}\overset{L}{\to}\boxed{c}}\overset{M}{\to} \qquad (1)$$

A strong monad T on $\mathbb{C}$ induces an arrow A_T by: $A_T(J,K) = \mathbb{C}(J,TK) = \mathcal{K}\ell(T)(J,K)$. Here $\mathcal{K}\ell(T)$ denotes the Kleisli category (see e.g. Moggi [21]). Prior to arrows, the notion of *Freyd category* is devised as another axiomatization of algebraic properties that are expected from "computations" [19, 23].

[1] The word "arrow" is reserved for Hughes' notion throughout the paper. An "arrow" in a category will be called a morphism or a 1-cell.

The latter notion of Freyd category come with a stronger categorical flavor; in Jacobs et al. [16] it is shown to be equivalent to the notion of arrow.

Remark 1.1 The previous arguments are true as long as we think of an arrow as carried by sets, with $A(J, K)$ being a set. This is our setting. However this is not an entirely satisfactory view in functional programming where one sees A as a type constructor—$A(J, K)$ should rather be an object of $\mathbb{C}$. In this case one can think of several variants of arrow and Freyd category. See Atkey [2]. The discussion later in the beginning of §5 is also relevant.

1.2 Arrow as Component Calculus

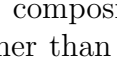
The current paper's goal is to settle *components* as *categorification* of computations, via (the algebraic theory of) arrows. Let us elaborate on this slogan.

A *component* here is in the sense of *component calculi*. Components are systems which, combined with one another by some component calculus, yield a bigger, more complicated system. This "divide-and-conquer" strategy brings order to design processes of large-scale systems that are otherwise messed up due to the very scale and complexity of the systems to be designed.

We follow the coalgebraic modeling of components in Barbosa [5]—which is also used in Hasuo et al. [11]—extending it later to an arrow-based modeling. In [5] a component is modeled as a coalgebra of the following type:

$$c \;:\; X \longrightarrow \big(T(X \times K)\big)^J \quad \text{in } \mathbf{Set}. \tag{2}$$

Here J is the set of possible input to the component; K is that of possible output; X is the set of (internal) states of the component which is a state-based machine; and T is a monad on **Set** that models the computational effect exhibited by the system. Overall, a coalgebraic component is a state-based system with specified input and output ports; it can be drawn as above on the right.

A crucial observation here is as follows. The notion of arrow in §1.1 is to axiomatize algebraic operators on computations as boxes—such as sequential composition  . Then, by regarding such boxes as components rather than as computations, we can employ the axiomatization of arrow as algebraic structure on components—a *component calculus*—with which one can compose components. The calculus is a basic one that allows embedding of pure functions, sequential composition and sideline. In fact in the second author's previous work [11] with Heunen, Jacobs and Sokolova, such algebraic operators on coalgebraic components (2) are defined and shown to satisfy the equational axioms.

1.3 Categorifying Computations into Components

Despite this similarity between computations and components, there is one level gap between them: from *sets* to *categories*. Let $\mathcal{A}(J, K)$ denote the collection of coalgebraic components like in (2), with input-type J, output-type K and fixed effect T, but with varying state spaces X. Then it is just natural to include morphisms between coalgebras in the overall picture, as behavior-preserving maps (see e.g. Rutten [24]) between components. Hence $\mathcal{A}(J, K)$ is now a *category*, specifically that of $\big(T(\_ \times K)\big)^J$-coalgebras. In contrast, with respect to computations there is no general notion of morphism between them, so the collection $A(J, K)$ of A-computations is a *set*.

This step of *categorification* [3] is not just for fun but in fact indispensable when we consider equational axioms. Later on we will concretely define the sequential composition $\xrightarrow{J} \boxed{c} \xrightarrow{K} \boxed{d} \xrightarrow{L}$ of coalgebraic components with matching I/O types; at this point we note that the state space of the composite is the product $X \times Y$ of the state space X of c and Y of d. Now let us turn to the axiom

$$(c \ggg d) \ggg e \;=\; c \ggg (d \ggg e) \;. \qquad (\ggg\text{-Assoc})$$

Denoting e's state space by U, the state space of the LHS is $(X \times Y) \times U$ while that of the RHS is $X \times (Y \times U)$. These are, as sets, not identical! Therefore the axiom can be at best satisfied up-to an isomorphism between components as coalgebras (and it is the case, see [11]). We note this phenomenon that the notion of satisfaction of equational axioms gets relaxed—from up-to equality to up-to an isomorphism—is typical with categorification [3].

This additional structure obtained through categorification, namely morphisms between components, has been further exploited in [11]. There it is shown that final coalgebras—the notion that only makes sense in presence of morphisms between coalgebras—form an arrow that is internal to the "arrow" of components, realizing an instance of the *microcosm principle* [4, 12]. An application of such nested algebraic structure (namely of arrows) is a *compositionality result*: the behavior of composed components can be computed from the behavior of each component.

We shall refer to the categorified notion of arrow—carried by components—as *categorical arrow*.

1.4 Lifting of Arrow Structure via Profunctors

To summarize: computations carry algebraic structure of an arrow; components carry a categorified version of it. The contribution of the current paper is to make the relationship between computations and components more direct. This is by developing the following scenario:

- given an arrow A,

- we define the notion of *(arrow-based) A-component* which generalizes Barbosa's modeling (2),
- and we show that these A-components carry categorical arrow structure that is in fact a lifting of the original arrow structure of A.

Therefore: we categorify A-computations to A-components.

A weaker version of this scenario has been already presented in [11]. However the last lifting part was obscured in details of direct calculations. What is novel in this paper is to work in **Prof**, the bicategory of profunctors. In fact, it is one theme of this paper to demonstrate use of calculations in **Prof**.

The starting point for this profunctor approach is [16]. There the $\mathsf{arr}, \ggg$-fragment of arrow (without first) is identified with a monoid in the category $[\mathbb{C}^{\mathrm{op}} \times \mathbb{C}, \mathbf{Set}]$ of bifunctors, where the latter is equipped with suitable monoidal structure. This means—in terms of profunctors that will be described in §2—that an arrow A (without first) is a *monad* in **Prof**, in an internal sense like in Street [25].

What really made our profunctor approach feasible was a further observation by the first author [1]. There the remaining first operator—whose mathematical nature was buried away in its dinaturality—is identified with a certain 2-cell in **Prof**. In fact, this 2-cell is a *strength* in an internal sense. Therefore an arrow (with its full set of operators, arr, $\ggg$ and first) is a *strong monad* in **Prof**. This observation pleasantly parallels the informal view of arrows as generalization of strong monads.

1.5 Organization of the Paper

In §2 we will introduce the necessary notions of dinatural transformation, (co)end and profunctor, in a rather leisurely pace. The two forms of the Yoneda lemma—the end- and coend-forms—are basic there. The materials there are essentially extracted from Kelly [17], which is a useful reference also in the current non-enriched (i.e. **Set**-enriched) setting. In §3 we follow [1, 16] and identify an arrow with an internal strong monad in **Prof**, setting **Prof** as our universe of discourse. In §4 we generalize Barbosa's coalgebraic components into arrow-based components. The main result—arrow-based components form a categorical arrow—is stated there. Its actual proof is in the subsequent §5 which is devoted to manipulation of 2-cells in **Prof**.

2 Categorical Preliminaries

2.1 End and Coend

In the sequel we shall often encounter a functor of the type $F : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{D}$, where a category $\mathbb{C}$ occurs twice with different variance. Given two such

$F, G : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{D}$, a *dinatural transformation* $\varphi : F \Rightarrow G$ consists of a family of morphisms in $\mathbb{D}$

$$\varphi_X \ : \ F(X,X) \longrightarrow G(X,X) \quad \text{for each } X \in \mathbb{C}$$

which is *dinatural*: for each morphism $f : X \to X'$ the following diagram commutes.

$$\begin{array}{ccccc} & \overset{F(f,X)}{\nearrow} F(X,X) & \xrightarrow{\varphi_X} & G(X,X) & \overset{G(X,f)}{\searrow} \\ F(X',X) & & & & G(X,X') \\ & \underset{F(X',f)}{\searrow} F(X',X') & \xrightarrow[\varphi_{X'}]{} & G(X',X') & \underset{G(f,X')}{\nearrow} \end{array} \tag{3}$$

Note the difference from a *natural transformation* $\psi : F \Rightarrow G$. The latter consists of a greater number of morphisms in $\mathbb{D}$: $\psi_{X,Y} : F(X,Y) \to G(X,Y)$ for each $X, Y \in \mathbb{C}$.

Two successive dinatural transformations $\varphi_1 : F_1 \Rightarrow F_2$ and $\varphi_2 : F_2 \Rightarrow F_3$ do not necessarily compose: dinaturality of each does not guarantee dinaturality of the obvious candidate of the composition $(\varphi_2 \circ \varphi_1)_X = (\varphi_2)_X \circ (\varphi_1)_X$. This makes it a tricky business to organize dinatural transformations in a categorical manner. Nevertheless, working with arrows, examples of dinaturality abound.

Dinaturality subsumes naturality: a natural transformation $\psi : F \Rightarrow G : \mathbb{C} \to \mathbb{D}$ can be thought of as a dinatural transformation, by presenting it as $\psi : F \circ \pi_2 \Rightarrow G \circ \pi_2 : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{D}$. Here $\pi_2 : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{C}$ is a projection.

(Co)end is the notion that is obtained by replacing naturality (for (co)cones) by dinaturality, in the definition of (co)limit. Precisely:

Definition 2.1 (End and coend) Let $\mathbb{C}, \mathbb{D}$ be categories and $F : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{D}$ be a functor.

- An *end* of F consists of an object $\int_{X \in \mathbb{C}} F(X,X)$ in $\mathbb{D}$ together with *projections*

  $$\pi_X : \left(\int_{X \in \mathbb{C}} F(X,X)\right) \longrightarrow F(X,X) \quad \text{for each } X \in \mathbb{C}$$

  such that, for each morphism $f : X \to X'$ in $\mathbb{C}$, the following diagram commutes.

  $$\begin{array}{ccccc} & \overset{\pi_{X'}}{\nearrow} F(X',X') & \xrightarrow{F(f,X')} & & \\ \int_X F(X,X) & & & & F(X,X') \\ & \underset{\pi_X}{\searrow} F(X,X) & \xrightarrow[F(X,f)]{} & & \end{array}$$

  In other words: the family $\{\pi_X\}_{X \in \mathbb{C}}$ forms a dinatural transformation from the constant functor $\Delta(\int_X F(X,X))$ to the functor F. An end is defined to be a universal one among such data: given an object $Y \in \mathbb{D}$ and a dinatural transformation $\varphi : \Delta Y \Rightarrow F$, there is a unique morphism $f : Y \to \int_X F(X,X)$ such that $\pi_X \circ f = \varphi_X$ for each $X \in \mathbb{C}$.

- A *coend* of F is a dual notion of an end. It consists of an object $\int^{X \in \mathbb{C}} F(X,X)$ in $\mathbb{D}$ together with *injections* $\iota_X : F(X,X) \to \int^X F(X,X)$ for each $X \in \mathbb{C}$. Its universality, together with that of an end, can be written as follows.

$$\frac{f : Y \longrightarrow \int_X F(X,X)}{\varphi_X : Y \to F(X,X)\,,\ \text{dinatural in } X} \qquad \frac{f : \int^X F(X,X) \longrightarrow Y}{\varphi_X : F(X,X) \to Y,\ \text{dinatural in } X}$$

(Co)ends need not exist; they do exist for example when $\mathbb{C}$ is small and $\mathbb{D}$ is (co)complete. See below.

The reader is referred to Mac Lane [20, Chap. IX] for more on (co)ends. Described there is the way to transform a functor $F : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{D}$ into $F^{\S} : \mathbb{C}^{\S} \to \mathbb{D}$, in such a way that the (co)end of F coincides with the (co)limit of $F^{\S}$. Therefore existence of (co)ends depends on the (co)completeness property of $\mathbb{D}$. In fact (co)end subsumes (co)limit, just as dinaturality subsumes naturality. Therefore a useful notational convention is to denote (co)limits also as (co)ends: for example $\mathrm{Colim}_X FX$ as $\int^X FX$.

Recalling the construction of any limit by a product and an equalizer [20, §V.2], an intuition about an end $\int_X F(X,X)$ is as follows: it is the product $\prod_X F(X,X)$ which is "cut down" so as to satisfy dinaturality. Dually, a coend $\int^X F(X,X)$ is the coproduct $\coprod_X F(X,X)$ quotiented modulo dinaturality.

2.2 Two Forms of the Yoneda Lemma

A typical example of an end arises as a set of (di)natural transformations. Given a small category $\mathbb{C}$ and functors $F, G : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$, we obtain a bifunctor

$$[F(+,-), G(-,+)] \ :\ \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \longrightarrow \mathbf{Set}\ , \quad (X,Y) \longmapsto [F(Y,X), G(X,Y)]\ . \tag{4}$$

Here $[S,T]$ denotes the set of functions from S to T, i.e. an exponential in **Set**. Note the variance: since $[-,+]$ is contravariant in its first argument, the variance of arguments of F is opposed in (4). Taking this functor (4) as F in Def. 2.1, we define an end $\int_X [F(X,X), G(X,X)]$. Such an end does exist when $\mathbb{C}$ is a small category, because **Set** has small limits (hence small ends).

Proposition 2.2 *Let us denote the set of dinatural transformations from F to G by* $\mathrm{Dinat}(F,G)$*. We have a canonical isomorphism in* **Set***:*

$$\mathrm{Dinat}(F,G) \xrightarrow{\cong} \int_X [F(X,X), G(X,X)]\ .$$

Proof It is due to the following correspondences.

$$\dfrac{\dfrac{1 \to \int_X [F(X,X), G(X,X)]}{1 \to [F(X,X), G(X,X)] \text{ dinatural in } X}\ (\dagger)}{F(X,X) \to G(X,X) \text{ dinatural in } X}\ (\ddagger)$$

Here (†) is by Def. 2.1; dinaturality is preserved along (‡) because of the naturality of Currying. □

The composite $\mathrm{Dinat}(F,G) \xrightarrow{\cong} \int_X [F(X,X), G(X,X)] \xrightarrow{\pi_X} [F(X,X), G(X,X)]$ carries a dinatural transformation φ to its X-component φ_X.

Since dinaturality subsumes naturality (§2.1), we have an immediate corollary:

Corollary 2.3 *Let $\mathbb{C}$ be a small category and $F, G : \mathbb{C} \to \mathbf{Set}$. By $\mathrm{Nat}(F,G)$ we denote the set of natural transformations $F \Rightarrow G$. We have*

$$\mathrm{Nat}(F,G) \xrightarrow{\cong} \int_X [FX, GX] \ . \qquad \square$$

The celebrated *Yoneda lemma* reduces the set $\mathrm{Nat}(\mathbb{C}(X,\_), F)$ of natural transformations into FX (see e.g. [6, 20]). Interpreted via Cor. 2.3, it yields:

Lemma 2.4 *(The Yoneda lemma, end-form) Given a small category $\mathbb{C}$ and a functor $F : \mathbb{C} \to \mathbf{Set}$, we have a canonical isomorphism*

$$\int_{X' \in \mathbb{C}} [\mathbb{C}(X,X'), FX'] \xrightarrow{\cong} FX \ . \qquad \square$$

The lemma becomes useful in the calculations below: it means an end on the LHS "cancels" with a hom-functor occurring in it.

From the end-form, we obtain the following coend-form. Its proof is straightforward but illuminating. It allows us to "cancel" a coend with a hom-functor inside it.

Lemma 2.5 *(The Yoneda lemma, coend-form) Given a small category $\mathbb{C}$ and a functor $F : \mathbb{C} \to \mathbf{Set}$, we have a canonical isomorphism*

$$\int^{X' \in \mathbb{C}} FX' \times \mathbb{C}(X', X) \xrightarrow{\cong} FX \ .$$

Proof We have the following canonical isomorphisms, for each $S \in \mathbf{Set}$.

$$\begin{aligned} \left[\int^{X'} FX' \times \mathbb{C}(X',X)\,,\ S \right] &\overset{\cong}{\Rightarrow} \int_{X'} \left[FX' \times \mathbb{C}(X',X)\,,\ S \right] && (\dagger) \\ &\overset{\cong}{\Rightarrow} \int_{X'} \left[\mathbb{C}(X',X)\,,\ [FX', S] \right] && \text{Currying} \\ &\overset{\cong}{\Rightarrow} [FX, S] && \text{the Yoneda lemma, end-form.} \end{aligned}$$

Here (†) is because the hom-functor $[\_\,, S]$ turns a colimit into a limit [20, §V.4], hence a coend into an end. Obviously the composite isomorphism is natural in S; therefore we have shown that

$$\mathbf{y}\bigl(\textstyle\int^{X'} \mathbb{C}(X', X) \times FX'\bigr) \xrightarrow{\cong} \mathbf{y}(FX) \quad : \mathbb{C} \longrightarrow \mathbf{Set}\ , \tag{5}$$

where $\mathbf{y} : \mathbb{C}^{\mathrm{op}} \to [\mathbb{C}, \mathbf{Set}]$ is the (contravariant) Yoneda embedding. By the Yoneda lemma the functor $\mathbf{y}$ is full and faithful; therefore it reflects isomorphisms. Hence (5) proves the claim. □

2.3 Profunctor

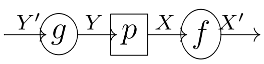
Definition 2.6 Let $\mathbb{C}$ and $\mathbb{D}$ be small categories. A *profunctor* P from $\mathbb{C}$ to $\mathbb{D}$ is a functor $P : \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$. It is denoted by $P : \mathbb{C} \nrightarrow \mathbb{D}$ (see on the right).

$$\frac{\mathbb{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathbb{D}}{\mathbb{D}^{\mathrm{op}} \times \mathbb{C} \longrightarrow \mathbf{Set}}$$

The notion of profunctor is also called *distributor*, *bimodule* or *module*. For more detailed treatment of profunctors see e.g. Benabou [7] and Borceux [9].

There are principally two ways to understand profunctors. One is as "generalized relations": profunctors are to functors what relations are to functions. The differences between a profunctor $P : \mathbb{C} \nrightarrow \mathbb{D}$ and a relation $R : S \nrightarrow T$ are as follows.

- A relation is two-valued: for each element $s \in S$ and $t \in T$, $R(s, t)$ is either empty (i.e. $(s, t) \notin R$) or filled (i.e. $(s, t) \in R$). In contrast, a profunctor is valued with arbitrary sets, that is, $P(Y, X) \in \mathbf{Set}$.
- The functoriality of a profunctor P induces *action* of morphisms in $\mathbb{C}$ and $\mathbb{D}$. For illustration let us depict an element $p \in P(Y, X)$ by a box $\xrightarrow{Y}\boxed{p}\xrightarrow{X}$. Given two morphisms $g : Y' \to Y$ in $\mathbb{D}$ and $f : X \to X'$ in $\mathbb{C}$, functoriality of P yields an element $P(g, f)(p) \in P(Y', X')$ (note the variance); the latter element is best depicted as follows.



(6)

The latter point motivates a different way of looking at profunctors: as generalized *modules* as in the theory of rings. These generalized modules are carried by a family of sets $\{P(Y, X)\}_{X \in \mathbb{C}, Y \in \mathbb{D}}$, with left-action of $\mathbb{C}$-arrows and right-action of $\mathbb{D}$-arrows. Also notice the similarity between (6) and the diagrams in §1 for computations/components. It is indeed this similarity that allows us to formalize arrows as certain profunctors (§3).

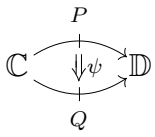
Definition 2.7 (Composition of profunctors) Given two successive profunctors $P : \mathbb{C} \nrightarrow \mathbb{D}$ and $Q : \mathbb{D} \nrightarrow \mathbb{E}$, their *composition* $Q \circ P : \mathbb{C} \nrightarrow \mathbb{E}$ is defined by the following coend. For $U \in \mathbb{E}$ and $X \in \mathbb{C}$,

$$(Q \circ P)(U, X) = \textstyle\int^{Y \in \mathbb{D}} Q(U, Y) \times P(Y, X)\ .$$

For profunctors as generalized relations, this composition operation corresponds to a *relational composition*: $(S \circ R)(x, z)$ if and only if $\exists y.\big(R(x, y) \wedge S(y, z)\big)$. For profunctors as modules, it corresponds to *tensor product* of modules. In any case, recall from §2.1 that the coend in Def. 2.7 is a coproduct $\coprod_Y Q(U, Y) \times P(Y, X)$—a bunch of pairs $(\ U \xrightarrow{} \boxed{q} \xrightarrow{} Y\ ,\ Y \xrightarrow{} \boxed{p} \xrightarrow{} X\ )$, with varying Y—quotiented modulo a certain equivalence $\simeq$. This equivalence $\simeq$ (dictated by dinaturality) intuitively says: the choice of intermediate $Y \in \mathbb{D}$ does not matter. Specifically, the equivalence $\simeq$ is generated by the following relation; here $f : Y \to Y'$ is a morphism in $\mathbb{D}$.

$$\big(\ U \xrightarrow{} \boxed{q} \xrightarrow{} Y \xrightarrow{} \textcircled{f} \xrightarrow{} Y'\ ,\ Y' \xrightarrow{} \boxed{p} \xrightarrow{} X\ \big) \ \simeq\ \big(\ U \xrightarrow{} \boxed{q} \xrightarrow{} Y\ ,\ Y \xrightarrow{} \textcircled{f} \xrightarrow{} Y' \xrightarrow{} \boxed{p} \xrightarrow{} X\ \big)\ .$$

An appropriate notion of *morphism* between parallel profunctors $P, Q : \mathbb{C} \nrightarrow \mathbb{D}$ is provided by a natural transformation $\psi : P \Rightarrow Q$, where P and Q are thought of as functors $P, Q : \mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$. All these data can be organized in a "2-categorical" manner as on the right. A problem now is that (horizontal) composition of 1-cells (i.e. profunctors) is not strictly associative: due to Def. 2.7 of composition by coends and products, associativity can be only ensured up-to coherent isomorphisms. The same goes for unitality; therefore profunctors form a *bicategory* (see [9]) instead of a 2-category.



Definition 2.8 (The bicategory **Prof**) The bicategory **Prof** has small categories as 0-cells, profunctors as 1-cells and natural transformations between them as 2-cells. The identity 1-cell $\mathbb{C} \nrightarrow \mathbb{C}$ is given by the hom-functor $\mathbb{C}(-, +) : \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$; it is the unit for composition because of the Yoneda lemma, coend-form (Lem. 2.5).

2.4 Some Properties of **Prof**

Here we describe some structural properties of **Prof** that will be exploited later, namely the direct image of a functor and tensor products in **Prof**. For the former, [7] is a principal reference; Fiore's notes [10] are not specifically on profunctors but provide useful insights into relevant mathematical concepts.

A function $f : S \to T$ induces the *direct image* relation $f_* : S \twoheadrightarrow T$, defined by: $f_*(s, t)$ iff $t = f(s)$. There is an analogous construction from functors to profunctors.

Definition 2.9 Let $F : \mathbb{C} \to \mathbb{D}$ be a functor between small categories. It gives rise to

$$\text{the } \textit{direct image} \text{ profunctor } \ F_* : \mathbb{C} \longrightarrow\!\!\!\!\!\rightarrow \mathbb{D} \qquad \text{by } F_*(Y, X) = \mathbb{D}(Y, FX)\ .$$

The mapping $(\_)_*$ also applies to natural transformations in an obvious way; this determines a *pseudo functor* (see e.g. [9]) $(\_)_* : \mathbf{Cat} \to \mathbf{Prof}$ that embeds **Cat** in **Prof**.

Notations 2.10 Throughout the rest of the paper, the direct image F_* of a functor F shall be simply denoted by F. The identity profunctor $\mathrm{id} : \mathbb{C} \nrightarrow \mathbb{C}$—that is the hom-functor—will be often denoted by $\mathbb{C} : \mathbb{C} \nrightarrow \mathbb{C}$.

The Cartesian product operator $\times$ in **Cat** lifts **Prof**; given profunctors $F : \mathbb{C} \nrightarrow \mathbb{C}'$ and $G : \mathbb{D} \nrightarrow \mathbb{D}'$, we define

$$F \times G : \mathbb{C} \times \mathbb{D} \nrightarrow \mathbb{C}' \times \mathbb{D}' \quad \text{by} \quad (F \times G)(X', Y', X, Y) = F(X', X) \times G(Y', Y) \ . \tag{7}$$

The symbol $\times$ occurring in the last denotes the Cartesian product in **Set**. The lifted operator $\times$ in **Prof** makes it a "monoidal bicategory," a notion whose precise definition involves delicate handling of coherence. We shall not do that in this paper. Nevertheless, we will need the following property.

Lemma 2.11 *The operation* $\times$ *on* **Prof** *is bifunctorial: that is, given four profunctors* $\mathbb{C} \overset{P}{\nrightarrow} \mathbb{D} \overset{Q}{\nrightarrow} \mathbb{E}$ *and* $\mathbb{C}' \overset{P'}{\nrightarrow} \mathbb{D}' \overset{Q'}{\nrightarrow} \mathbb{E}'$ *we have* $(Q \circ P) \times (Q' \circ P') \overset{\cong}{\Rightarrow} (Q \times Q') \circ (P \times P')$.

Proof This is due to the Fubini theorem for coends. See [20, §IX.8] □

It is obvious that the operator $\times$ acts also on 2-cells (that are natural transformations).

3 Arrows as Profunctors

We review the results in [1, 16] that identify Hughes' notion of arrow with a profunctor with additional algebraic structure.

First we present the precise definition of arrow. Usually it is defined over a Cartesian category $\mathbb{C}$. However, since it is rather the monoidal structure of $\mathbb{C}$ that is essential, we shall work with a monoidal category.

Definition 3.1 (Arrow [13]) Given a monoidal category $\mathbb{C} = (\mathbb{C}, \otimes, I)$, an *arrow* over $\mathbb{C}$ consists of carrier sets $\{A(J, K)\}_{J,K \in \mathbb{C}}$ and operators arr, $\ggg$ and first as described in §1.1. The operators must satisfy the following equational

axioms.

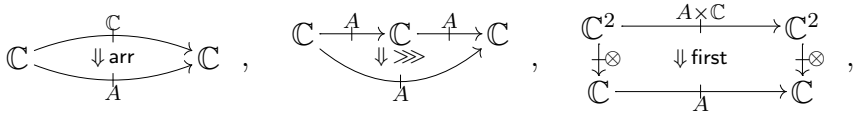
$$
\begin{aligned}
(a \ggg b) \ggg c &= a \ggg (b \ggg c) && (\ggg\text{-Assoc})\\
\mathsf{arr}\,(g \circ f) &= \mathsf{arr}\, f \ggg \mathsf{arr}\, g && (\mathsf{arr}\text{-Func1})\\
\mathsf{arr}\,\mathrm{id}_J \ggg_{J,J,K} a &= a = a \ggg_{J,K,K} \mathsf{arr}\,\mathrm{id}_K && (\mathsf{arr}\text{-Func2})\\
\mathsf{first}_{J,K,I}\, a \ggg \mathsf{arr}\,\rho_K &= \mathsf{arr}\,\rho_K \ggg a && (\rho\text{-Nat})\\
\mathsf{first}_{J,K,L}\, a \ggg \mathsf{arr}(\mathrm{id}_K \otimes f) &= \mathsf{arr}(\mathrm{id}_J \otimes f) \ggg \mathsf{first}_{J,K,M}\, a && (\mathsf{arr}\text{-Centr})\\
(\mathsf{first}_{J,K,L\otimes M}\, a) \ggg (\mathsf{arr}\,\alpha_{K,L,M}) &= (\mathsf{arr}\,\alpha_{J,L,M}) \ggg \mathsf{first}(\mathsf{first}\, a) && (\alpha\text{-Nat})\\
\mathsf{first}_{J,K,L}(\mathsf{arr}\, f) &= \mathsf{arr}(f \otimes \mathrm{id}_L) && (\mathsf{arr}\text{-Premon})\\
\mathsf{first}_{J,L,M}(a \ggg b) &= (\mathsf{first}_{J,K,M}\, a) \ggg (\mathsf{first}_{K,L,M}\, b) && (\mathsf{first}\text{-Func})
\end{aligned}
$$

Here some subscripts are suppressed. The morphism $\rho_K : K \otimes I \xrightarrow{\cong} K$ is the right unitor isomorphism; α denotes an associator isomorphism. The names of the axioms hint their correspondence to the (premonoidal) structure of *Freyd categories* [19, 23].

Next we introduce the corresponding construct in **Prof**, which we shall tentatively call a **Prof**-*arrow*.

Definition 3.2 Let $\mathbb{C} = (\mathbb{C}, \otimes, I)$ be a small monoidal category. A **Prof**-*arrow* over $\mathbb{C}$ is:

- a profunctor $A : \mathbb{C} \nrightarrow \mathbb{C}$,
- equipped with natural transformations $\mathsf{arr}, \ggg, \mathsf{first}$ of the following types:



  where all the diagrams are in **Prof**,
- subject to the equalities in Table 1. Recall Notations 2.10; for example the profunctor $\langle \mathbb{C}, I \rangle$ in (first-ρ) is the functor $\langle \mathbb{C}, I \rangle : X \mapsto (X, I)$, embedded in **Prof** by taking its direct image.

The notion of **Prof**-arrow is in fact a familiar one: it is an internal *strong monad* in **Prof**. Indeed, when one draws the same 2-cells in **Cat** instead of in **Prof**—replacing A by T, arr by η^T, $\ggg$ by μ^T and first by str'—the definition coincides with that of strong monad [18, 21]. [2] More specifically, the first two axioms in Table 1 are for the monad laws; and the remaining axioms asserts compatibility of strength with monoidal and monad structure. For example, the axiom (first-$\ggg$) interpreted in **Cat** is read as the commutativity of the

[2] The corresponding strength operator str' is of the type $\mathsf{str}' : TX \otimes Y \to T(X \otimes Y)$, which is slightly different from the usual strength operator that is $\mathsf{str} : X \otimes TY \to T(X \otimes Y)$.

following diagram.

$$\begin{array}{ccccc} T^2X \otimes Y & \xrightarrow{\mathsf{str}'} & T(TX \otimes Y) & \xrightarrow{T\mathsf{str}'} & T^2(X \otimes Y) \\ {\scriptstyle \mu^T \otimes Y}\downarrow & & & & \downarrow{\scriptstyle \mu^T} \\ TX \otimes Y & & \xrightarrow[\mathsf{str}']{} & & T(X \otimes Y) \end{array}$$

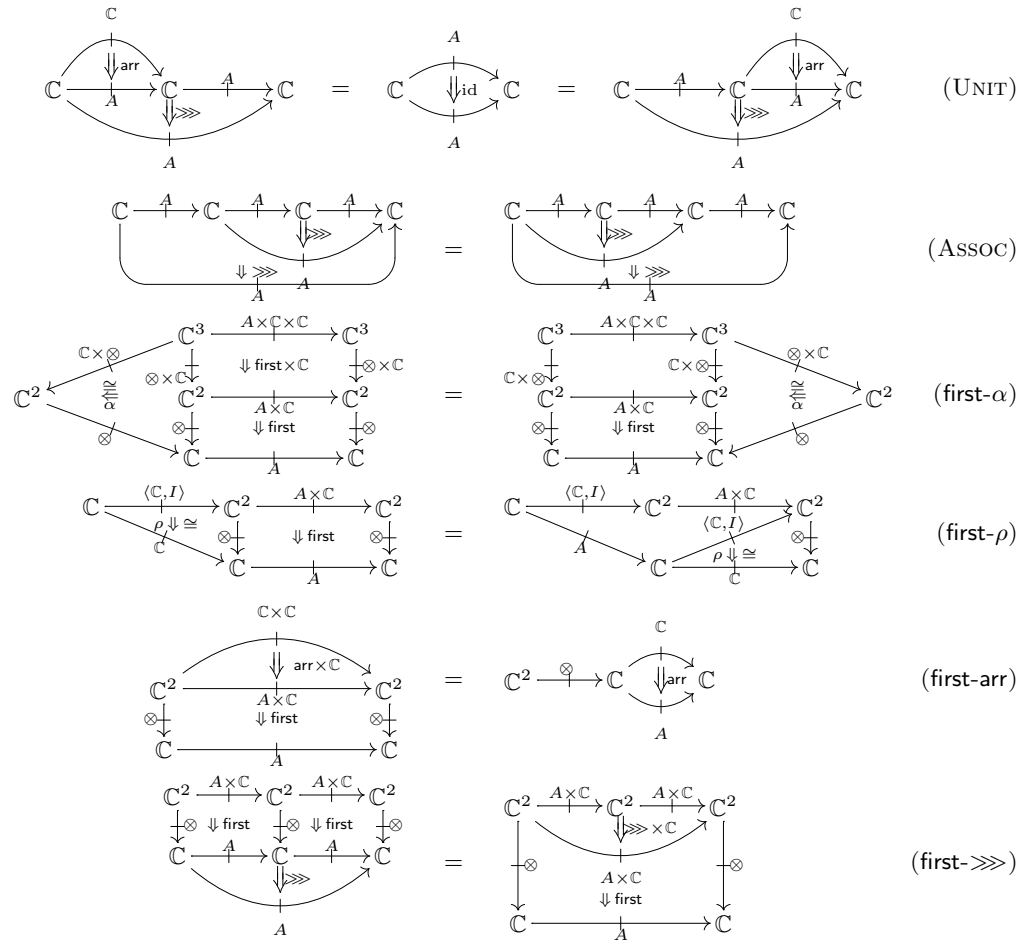


Table 1
Equational axioms for **Prof**-arrow

Proposition 3.3 *[1] For a monoidal category $\mathbb{C}$ that is small, the notion of arrow (Def. 3.1) and that of* **Prof***-arrow (Def. 3.2) are equivalent.*

Proof While the reader is referred to [1] for a detailed proof, we shall illustrate a few highlights in the correspondence between the two notions. We shall write arr', $\ggg'$ and first' (with primes) for the three operators of a **Prof**-arrow (Def. 3.2), to distinguish them from the corresponding operators of an arrow (Def. 3.1).

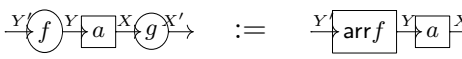
Let us first observe that a 2-cell first' in **Prof** gives rise to the first operator

in Def. 3.1. The former is an element of the LHS below, where $\ggg$ denotes composition of profunctors (Def. 2.7).

$$
\begin{aligned}
&\mathrm{Nat}\big(\ (\otimes \circ (A \times \mathbb{C}))(-, +_1, +_2)\,,\ (A \circ \otimes)(-, +_1, +_2)\ \big) \\
&\cong \textstyle\int_{X,K,Y \in \mathbb{C}} \big[\ (\otimes \circ (A \times \mathbb{C}))(X, K, Y)\,,\ (A \circ \otimes)(X, K, Y)\ \big] && \text{by Cor. 2.3} \\
&\cong \textstyle\int_{X,K,Y} \big[\ \int^{J,L} \mathbb{C}(X, J \otimes L) \times A(J, K) \times \mathbb{C}(L, Y)\,, \\
&\qquad\qquad \textstyle\int^{U} A(X, U) \times \mathbb{C}(U, K \otimes Y)\ \big] && \text{by Def. 2.7, Def. 2.9 and (7)} \\
&\cong \textstyle\int_{X,K,Y,J,L} \big[\ \mathbb{C}(X, J \otimes L) \times A(J, K) \times \mathbb{C}(L, Y)\,,\ \int^{U} A(X, U) \times \mathbb{C}(U, K \otimes Y)\ \big] \\
&\qquad\qquad \text{since a hom-functor } [-, S] \text{ turns a coend into an end} \\
&\cong \textstyle\int_{X,K,Y,J,L} \big[\ \mathbb{C}(X, J \otimes L),\ \big[\ A(J, K),\ \big[\ \mathbb{C}(L, Y)\,, \\
&\qquad\qquad \textstyle\int^{U} A(X, U) \times \mathbb{C}(U, K \otimes Y)\ \big]\ \big]\ \big] && \text{by Currying} \\
&\cong \textstyle\int_{J,K,L} \big[\ A(J, K),\ A(J \otimes L, K \otimes L)\ \big] \\
&\qquad\qquad \text{by canceling } X, Y \text{ by Lem. 2.4 and } U \text{ by Lem. 2.5} \\
&\cong \mathrm{Nat}_{J,K}\mathrm{Dinat}_{L}\big(\ A(J, K),\ A(J \otimes L, K \otimes L)\ \big) && \text{by Prop. 2.2 and Cor. 2.3.}
\end{aligned}
$$

Therefore a 2-cell first' in **Prof** gives rise to a family of functions $A(J, K) \to A(J \otimes L, K \otimes L)$ that is natural in J, K and dinatural in L. This is precisely the type of the first operator in Def. 3.1. The equational axioms of an arrow are indeed satisfied due to those of a **Prof**-arrow. We note that the axiom (arr-Centr) is satisfied not because of any specific axiom of a **Prof**-arrow, but because of the dinaturality of first' as a 2-cell in **Prof**.

For the reverse direction where an arrow induces a **Prof**-arrow, we have to equip the carrier $\{A(J, K)\}_{J,K}$ of an arrow with action of morphisms in $\mathbb{C}$, rendering A into a functor $\mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Set}$. This is done with the help of arrow operators. Specifically, $A(g, f)(a) := \mathsf{arr}f \ggg a \ggg \mathsf{arr}g$, that is:



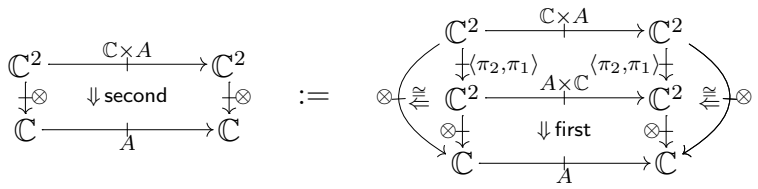
Each of the arrow operators yield its corresponding **Prof**-arrow operator; the latter's (di)naturality is derived from the arrow axioms. So are the equational axioms for a **Prof**-arrow. □

Prop. 3.3 offers a novel mathematical understanding of the notion of arrow. Its axiomatization seems to have stronger justifications than the original one (Def. 3.1) does. It also seems simpler than the treatment of first in Freyd categories which involves technicalities like premonoidal categories and central morphisms. It is this simplicity that is exploited in the rest of the paper.

When the base monoidal category $\mathbb{C}$ is symmetric—which is our setting in the sequel—we can obtain another sideline operator second.

Definition 3.4 Let A be an arrow over a small symmetric monoidal category

(SMC) $\mathbb{C}$. We define an extra operator second as the following 2-cell in **Prof**.



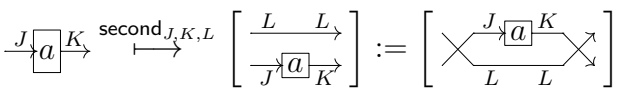
(8)

Here the profunctor $\langle\pi_2, \pi_1\rangle$ is the direct image of the functor $\langle\pi_2, \pi_1\rangle : \mathbb{C}^2 \to \mathbb{C}^2$, mapping (X, Y) to (Y, X) (cf. Notations 2.10).

Notations 3.5 In the above diagrams as well as elsewhere, there appear two different classes of iso 2-cells in **Prof**. One class is due to the unitality/associativity/symmetry of $\otimes$ on a monoidal base category $\mathbb{C}$; they are iso 2-cells in **Cat** embedded in **Prof** via direct image (§2.4). Such iso 2-cells shall be filled explicitly with the $\cong$ sign, like the two on the RHS in (8).

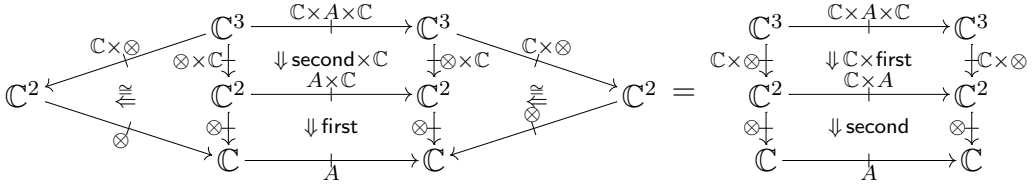
The other class is due to the properties of the operation $\times$ on **Prof**, typically Lem. 2.11. Such iso 2-cells will be denoted by empty polygons, like the one on the RHS in (8).

Some calculations like in the proof of Prop. 3.3 reveal that this new operator realizes a class of functions $A(J, K) \xrightarrow{\mathsf{second}_{J,K,L}} A(L \times J, L \times K)$, that is graphically



.

Lemma 3.6 *Between the* first *and* second *operators, the following equality holds.*



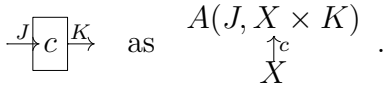
Proof Use the equality (first-α) and the coherence for an SMC $\mathbb{C}$. □

4 Arrow-Based Components

In this section we develop the scenario in §1.4 in technical terms. First we introduce an arrow-based coalgebraic modeling of components.

Definition 4.1 (A-component) Let A be an arrow on **Set**, and $J, K \in$ **Set**. An *(arrow-based) A-component* with input-type J, output-type K and *computational structure* A is a coalgebra for the functor $A(J, \_ \times K) :$ **Set** $\to$ **Set**.

That is,



Here an arrow A is in the sense of Def. 3.1. There the base $\mathbb{C}$ of an arrow need not be small; thus we choose $(\mathbf{Set}, \times, 1)$ as $\mathbb{C}$. Our modeling specializes to Barbosa's (2) when we take as A a monad-based arrow A_T (§1.1). Our modeling not only generalizes Barbosa's one but also brings conceptual clarity to the subsequent arguments.

Our goal is to lift the arrow structure of A to the categorical arrow structure of A-components. Let us make this goal precise.

Definition 4.2 (Categorical arrow) A *categorical arrow* consists of

- a family $\{\mathcal{A}(J,K)\}_{J,K}$ of *carrier* categories indexed by $J, K \in \mathbf{Set}$;
- (interpretation of) arrow operators arr, $\ggg$ and first (cf. Def. 3.1), namely functors

$$\begin{array}{ccccl} \mathbf{1} & \xrightarrow{\mathsf{arr}\, f} & \mathcal{A}(J,K) & & \text{for each function } f : J \to K \text{ in } \mathbf{Set}, \\ \mathcal{A}(J,K) \times \mathcal{A}(K,L) & \xrightarrow{\ggg_{J,K,L}} & \mathcal{A}(J,L) & & \text{for each } J,K,L \in \mathbf{Set}, \\ \mathcal{A}(J,K) & \xrightarrow{\mathsf{first}_{J,K,L}} & \mathcal{A}(J \times L, K \times L) & & \text{for each } J,K,L \in \mathbf{Set}. \end{array}$$

  Here the category $\mathbf{1}$ is the one-object and one-arrow (i.e. terminal) category; and
- the operators are subject to the arrow axioms in Def. 3.1, up-to isomorphisms. For example, as to the axiom ($\ggg$-Assoc), the following diagram must commute up-to an isomorphism.

$$\begin{array}{ccc} \mathcal{A}(J,K) \times \mathcal{A}(K,L) \times \mathcal{A}(L,M) & \xrightarrow{\ggg_{J,K,L} \times \mathrm{id}} & \mathcal{A}(J,L) \times \mathcal{A}(L,M) \\ {\scriptstyle \mathrm{id} \times \ggg_{K,L,M}}\downarrow & \Downarrow \cong & \downarrow{\scriptstyle \ggg_{J,L,M}} \\ \mathcal{A}(J,K) \times \mathcal{A}(K,M) & \xrightarrow[\ggg_{J,K,M}]{} & \mathcal{A}(J,M) \end{array} \tag{9}$$

The graphical understanding of a categorical arrow is the same as that of an arrow; see §1.1. In §1.3 we described why it is natural and necessary to require the axioms be satisfied only up-to isomorphisms.

Remark 4.3 Satisfaction up-to isomorphisms raises a *coherence* issue. The precise coherence condition for categorical arrows is described in [11], in a more general form of coherence for categorical models of FP-theories. Although we shall not further discuss the coherence issue, the calculations later in §5 provide us a much better grip on it than the direct calculations in [11] do.

The notion of categorical arrow in Def. 4.2 could be formalized on any monoidal category $\mathbb{C}$ other than $\mathbf{Set}$, although we do not need such additional generality.

The main contribution of this paper is the following result as well as its proof presented using the rest of the paper.

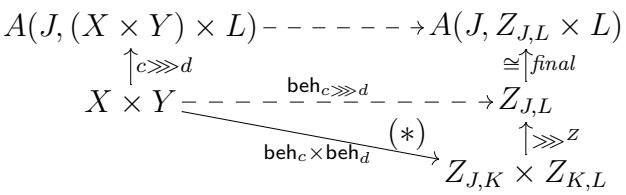
Theorem 4.4 *(Main contribution) Let A be an arrow on* **Set**. *The categories $\{\,\mathbf{Coalg}(A(J, \_ \times K))\}_{J,K}$ of A-components carry a categorical arrow.*

On top of it, we can appeal to the formalization [11, 12] of the *microcosm principle* [4] to obtain the following *compositionality result*.

Corollary 4.5 *In the setting of Thm. 4.4, assume further that for each $J, K \in$* **Set** *the functor $A(J, \_ \times K)$ has a final coalgebra $\zeta_{J,K} : Z_{J,K} \xrightarrow{\cong} A(J, Z_{J,K} \times K)$.*

(i) *The family $\{Z_{J,K}\}_{J,K}$ is canonically an arrow.*

(ii) *Behaviors by coinduction are compositional with respect to arrow operators. For example, with respect to the operator $\ggg$, this means the following. Given two A-components $c : X \to A(J, X \times K)$ and $d : Y \to A(K, Y \times L)$ with matching I/O types, the triangle $(*)$ below commutes.*



Here $c \ggg d$ is "composition of components" using the categorical arrow structure in Thm. 4.4; $\ggg^Z$ is "composition of behaviors" derived in (i); and $\mathsf{beh}_{c \ggg d}$ is the behavior map for the composed components induced by coinduction (the square on the top). □

In [11, 12] it is shown that algebraic structure carried by the categories of coalgebras—like the one in Thm 4.4—can be obtained by:

- the same structure on the base categories, and
- the *lax compatibility* of the signature functors with the relevant algebraic structure.

In this case the algebraic structure on the base categories lifts to the categories of coalgebras. We shall follow this path. Restricting the general definitions and results in [11, 12] to the current setting, we obtain the following.

Definition 4.6 Let $\{F_{J,K} : \mathbf{Set} \to \mathbf{Set}\}_{J,K}$ be a family of endofunctors, indexed by $J, K \in$ **Set**. It is said to be a *lax arrow functor* if:

- it is equipped with the following natural transformations

$$\begin{aligned} F_{\mathsf{arr} f} &: 1 \longrightarrow F_{J,K}1 \ , \\ F_{\ggg_{J,K,L}} &: F_{J,K}X \times F_{K,L}Y \longrightarrow F_{J,L}(X \times Y) \ , \\ F_{\mathsf{first}_{J,K,L}} &: F_{J,K}X \longrightarrow F_{J \times L, K \times L}X \ , \end{aligned}$$

each of which is natural in X, Y, for each $J, K, L \in \mathbf{Set}$ and each $f : J \to K$ in $\mathbf{Set}$;

- that are subject to the equations in Table 2, that are parallel to those in Def. 3.1. The diagrams there are all in $\mathbf{Set}$; obvious subscripts are suppressed.

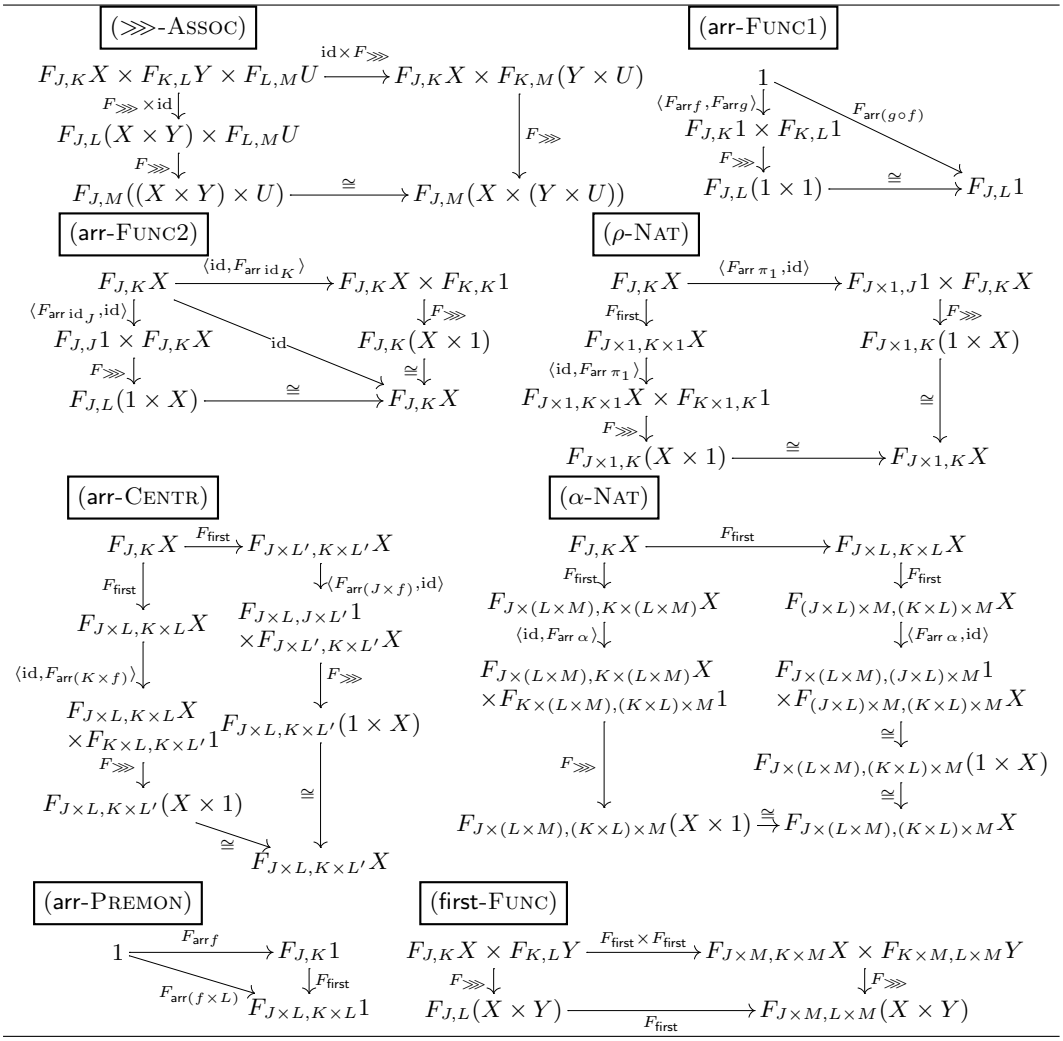


Table 2
Equational axioms for lax arrow functors

A lax arrow functor therefore looks like an arrow (think of $F_{J,K}(X)$ in place of $A(J,K)$), but it carries an extra parameter (like X, Y or $X \times Y$) around.

Proposition 4.7 *If $\{F_{J,K}\}_{J,K}$ is a lax arrow functor, then $\{\mathbf{Coalg}(F_{J,K})\}_{J,K}$ is canonically a categorical arrow.*

Proof This follows from a general result like [11, Thm. 4.6]. Here we shall briefly illustrate what the categorical arrow $\{\mathbf{Coalg}(F_{J,K})\}_{J,K}$ looks like, by

describing the sequential composition $\ggg : \mathbf{Coalg}(F_{J,K}) \times \mathbf{Coalg}(F_{K,L}) \longrightarrow \mathbf{Coalg}(F_{J,L})$. Using $F_{\ggg}$ in Def. 4.6 it is defined as follows.

$$\begin{pmatrix} F_{J,K}X \\ \uparrow^{c} \\ X \end{pmatrix}, \begin{pmatrix} F_{K,L}Y \\ \uparrow^{d} \\ Y \end{pmatrix} \overset{\ggg}{\longmapsto} \begin{array}{c} F_{J,L}(X \times Y) \\ \uparrow^{F_{\ggg}} \\ F_{J,K}X \times F_{K,L}Y \\ \uparrow^{c \times d} \\ X \times Y \end{array}$$

The definitions are similar for the other arrow operators. The arrow axioms are satisfied due to the corresponding equational condition on the lax arrow functor. □

This proposition reduces our goal (Thm. 4.4) to showing that the family $\{A(J, \_ \times K)\}_{J,K}$ is a lax arrow functor. This is what will be shown in the next section, through manipulation of 2-cells in **Prof**.

5 Calculations in Prof

There is one technical issue in front of us: the size issue. The 0-cells of **Prof** are *small* categories; the smallness restriction is necessary for composition of profunctors to be well-defined (Def. 2.7). However, with **Set** being not small, the arrow A in Def. 4.1 cannot be a 1-cell in **Prof**. The arrow A needs to be based on **Set** so that $A(J, \_ \times K)$ is an endofunctor $\mathbf{Set} \to \mathbf{Set}$.

In this paper we shall get round of the problem by pretending that **Set** is small. There are two possible justifications.

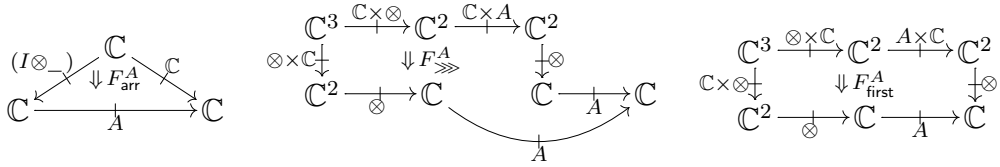
- We can resort to the category **Ens** of classes when it is needed—such as when we take composition of profunctors via a coend. This means upgrading all the sizes that appear in the definition of **Prof**: its 0-cells are locally small categories; its 1-cells $P : \mathbb{C} \nrightarrow \mathbb{D}$ are bifunctors $\mathbb{D}^{\mathrm{op}} \times \mathbb{C} \to \mathbf{Ens}$. In this case, in Def. 4.1, we would restrict the arrow A to be *small*, in the sense that its image $A(J, K)$ restricts to **Set**. More detailed treatment is found in [1].

$$\begin{array}{ccc} \mathbf{Set}^{\mathrm{op}} \times \mathbf{Set} & \xrightarrow{\quad A \quad} & \mathbf{Ens} \\ & \searrow & \uparrow \\ & & \mathbf{Set} \end{array}$$

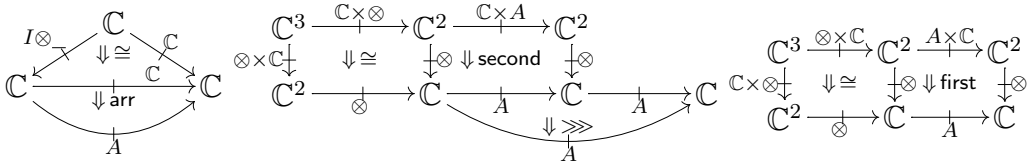
- We replace **Set** by some small cocomplete category defined internally in a suitable topos [14]. In other words, we develop our theory on top of a certain type theory which is modeled by such a topos.

In any case, we would like to isolate the size issue as much as possible. Therefore we shall first establish those technical results which hold for any small symmetric monoidal category $(\mathbb{C}, \otimes, I)$. These results are proved by manipulating 2-cells in **Prof**. After that we instantiate $(\mathbb{C}, \otimes, I)$ by $(\mathbf{Set}, \times, 1)$—pretending that **Set** is small.

Definition 5.1 Let $(\mathbb{C}, \otimes, I)$ be a small SMC, and A be an arrow on it. There arise three 2-cells in **Prof**—which we denote by F^A_{arr}, $F^A_{\ggg}$ and F^A_{first}—of the following types.



Explicitly, these 2-cells are given by the following composites.



Here the 1-cell $I \otimes \_$ on the left is the direct image of the functor $X \mapsto I \otimes X$ (Notations 2.10); recall that I denotes the monoidal unit. Also recall Notations 3.5. The 2-cells $\mathsf{arr}, \ggg, \mathsf{first}, \mathsf{second}$ are due to the arrow structure of A (Def. 3.2, 3.4).

The motivation for this definition is clear from the names of the 2-cells. Indeed, through some calculations in **Prof** and application of the Yoneda lemma, one easily sees that the three 2-cells $F^A_{\mathsf{arr}}, F^A_{\ggg}, F^A_{\mathsf{first}}$ are the same thing as (di)natural transformations

$$
\begin{aligned}
F^A_{\mathsf{arr}} \;&:\; \mathbb{C}(J,K) \longrightarrow A(J, I \otimes K) \;, \quad \text{natural in } J, K; \\
F^A_{\ggg}{}_{J,K,L} \;&:\; A(J, X \otimes K) \times A(K, Y \otimes L) \longrightarrow A(J, (X \otimes Y) \otimes L) \;, \\
&\qquad\qquad \text{natural in } J, L, X, Y, \text{ dinatural in } K, \\
F^A_{\mathsf{first}}{}_{J,K,L} \;&:\; A(J, X \otimes K) \longrightarrow A(J \otimes L, X \otimes (K \otimes L)) \;, \\
&\qquad\qquad \text{natural in } J, K, X, \text{ dinatural in } L,
\end{aligned}
$$

respectively. These (di)natural transformations bear clear similarity to the ones in Def. 4.6 when $F_{J,K}$ is instantiated with $A(J, \_ \otimes K)$.

Let us now turn to equations.

Lemma 5.2 *Let A be an arrow over a small SMC $\mathbb{C}$. The three 2-cells $F^A_{\mathsf{arr}}, F^A_{\ggg}$ and F^A_{first} in Def. 5.1 satisfy the equalities in Table 3; they are parallel to the equalities in Def. 3.2.*

Proof First expand the definitions of $F^A_{\mathsf{arr}}, F^A_{\ggg}$ and F^A_{first}, and then use the equational axioms in Def. 3.2. One also needs Lem. 3.6. □

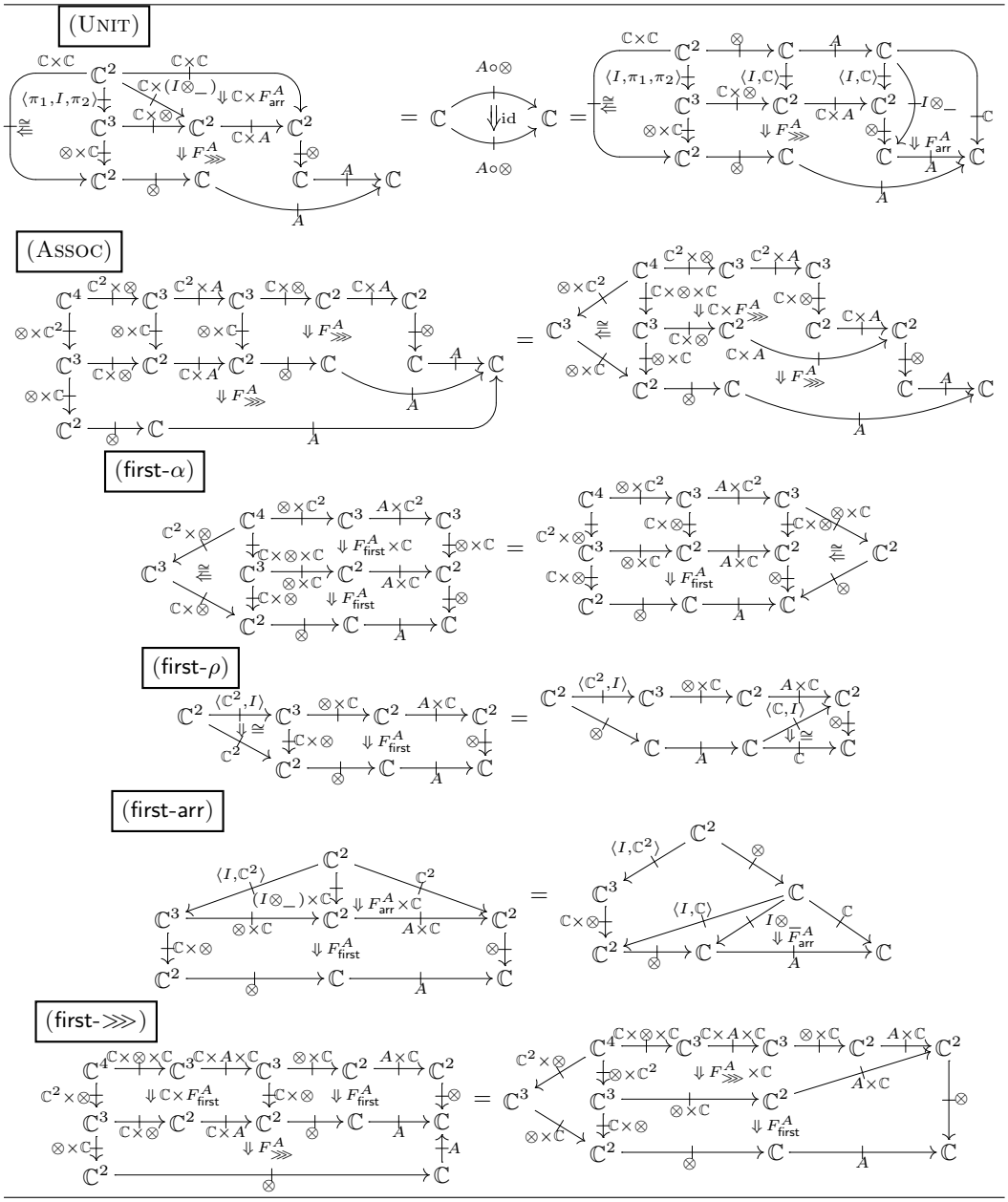


Table 3
Equalities that hold for $F^A_{\mathsf{arr}}, F^A_{\ggg}, F^A_{\mathsf{first}}$

The equalities in Table 3 might look complicated. However, coming up with them is rather routine work looking at Def. 5.1 and Def. 3.2.

We now instantiate $(\mathbb{C}, \otimes, I)$ with $(\mathbf{Set}, \times, 1)$, pretending **Set** to be small.

Lemma 5.3 *Let A be an arrow $\mathbf{Set}^{\mathrm{op}} \times \mathbf{Set} \to \mathbf{Set}$. The family $\{A(J, \_ \times K)\}_{J,K}$ of endofunctors is a lax arrow functor.*

Proof The three 2-cells in Def. 5.1 provide the three natural transformations required in Def. 4.6. The equations asserted in Def. 4.6 follow from those in Lem. 5.2. Checking all this is (laborious) routine work. □

Combining Prop. 4.7 and Lem. 5.3, our main result Thm. 4.4 is proved.

Remark 5.4 A characterization of categorical arrows in the spirit of Prop. 3.3 can possibly yield a even more direct proof of Thm. 4.4. Unfortunately until now we lack necessary infrastructure such as a lifting result like Prop. 4.7. We are currently investigating possible formalization using fibered spans (see e.g. Jacobs [15]).

In **Prof** the *trace* operator for an arrow (`loop` in Paterson [22], see also Benton and Hyland [8]) can be formalized in a similar way to other operators like $\ggg$. Its description as well as possible application to components will presented in another venue.

Acknowledgments

Thanks are due to Paul-André Melliès for advocating use of profunctors; and to Marcelo Fiore, Bart Jacobs and Bartek Klin for helpful discussions.

References

[1] Asada, K., *Arrows are strong monads* (2009), preprint, `www.kurims.kyoto-u.ac.jp/~asada/papers/arrStrMnd.pdf`.

[2] Atkey, R., *What is a categorical model of arrows?*, in: V. Capretta and C. McBride, editors, *Mathematically Structured Functional Programming*, 2008.

[3] Baez, J. C. and J. Dolan, *Categorification*, Contemp. Math. **230** (1998), pp. 1–36.

[4] Baez, J. C. and J. Dolan, *Higher dimensional algebra III: n-categories and the algebra of opetopes*, Adv. Math **135** (1998), pp. 145–206.
URL `citeseer.ist.psu.edu/article/baez97higherdimensional.html`

[5] Barbosa, L., "Components as Coalgebras," Ph.D. thesis, Univ. Minho (2001).

[6] Barr, M. and C. Wells, "Toposes, Triples and Theories," Springer, Berlin, 1985, available online.

[7] Bénabou, J., *Distributors at work*, Lecture notes taken by T. Streicher (2000), `www.mathematik.tu-darmstadt.de/~streicher/FIBR/DiWo.pdf.gz`.

[8] Benton, N. and M. Hyland, *Traced premonoidal categories*, Theoretical Informatics and Applications **37** (2003), pp. 273–299.

[9] Borceux, F., "Handbook of Categorical Algebra," Encyclopedia of Mathematics **50, 51 and 52**, Cambridge Univ. Press, 1994.

[10] Fiore, M., *Rough notes on presheaves* (2001), available online.

[11] Hasuo, I., C. Heunen, B. Jacobs and A. Sokolova, *Coalgebraic components in a many-sorted microcosm*, in: A. Kurz, M. Lenisa and A. Tarlecki, editors, *CALCO*, Lect. Notes Comp. Sci. **5728** (2009), pp. 64–80.

[12] Hasuo, I., B. Jacobs and A. Sokolova, *The microcosm principle and concurrency in coalgebra*, in: *Foundations of Software Science and Computation Structures*, Lect. Notes Comp. Sci. **4962** (2008), pp. 246–260.

[13] Hughes, J., *Generalising monads to arrows.*, Science of Comput. Progr. **37** (2000), pp. 67–111.

[14] Hyland, J. M. E., *A small complete category*, Ann. Pure & Appl. Logic **40** (1988), pp. 135–165.

[15] Jacobs, B., "Categorical Logic and Type Theory," North Holland, Amsterdam, 1999.

[16] Jacobs, B., C. Heunen and I. Hasuo, *Categorical semantics for arrows*, J. Funct. Progr. **19** (2009), pp. 403–438.

[17] Kelly, G. M., "Basic Concepts of Enriched Category Theory," Number 64 in LMS, Cambridge Univ. Press, 1982, available online: `http://www.tac.mta.ca/tac/reprints/articles/10/tr10abs.html`.

[18] Kock, A., *Monads on symmetric monoidal closed categories*, Arch. Math. **XXI** (1970), pp. 1–10.

[19] Levy, P. B., A. J. Power and H. Thielecke, *Modelling environments in call-by-value programming languages*, Inf. & Comp. **185** (2003), pp. 182–210.

[20] Mac Lane, S., "Categories for the Working Mathematician," Springer, Berlin, 1998, 2nd edition.

[21] Moggi, E., *Notions of computation and monads*, Inf. & Comp. **93(1)** (1991), pp. 55–92.

[22] Paterson, R., *A new notation for arrows*, in: *ICFP*, 2001, pp. 229–240.

[23] Power, J. and E. Robinson, *Premonoidal categories and notions of computation.*, Math. Struct. in Comp. Sci. **7** (1997), pp. 453–468.

[24] Rutten, J. J. M. M., *Universal coalgebra: a theory of systems*, Theor. Comp. Sci. **249** (2000), pp. 3–80.

[25] Street, R., *The formal theory of monads*, Journ. of Pure & Appl. Algebra **2** (1972), pp. 149–169.

[26] Uustalu, T. and V. Vene, *Comonadic notions of computation*, Elect. Notes in Theor. Comp. Sci. **203** (2008), pp. 263–284.

On Coalgebras over Algebras

Adriana Balan[1]

Department of Mathematics II
University Politehnica of Bucharest

Alexander Kurz

Department of Computer Science
University of Leicester

Abstract

We extend Barr's well-known characterization of the final coalgebra of a *Set*-endofunctor as the completion of its initial algebra to the Eilenberg-Moore category of algebras for a *Set*-monad $\mathbf{M}$ for functors arising as liftings. As an application we introduce the notion of commuting pair of endofunctors with respect to the monad $\mathbf{M}$ and show that under reasonable assumptions, the final coalgebra of one of the endofunctors involved can be obtained as the free algebra generated by the initial algebra of the second endofunctor.

Keywords: Coalgebra, algebras over a monad

1 Introduction

Although most research on coalgebras is focused on *Set*-coalgebras, coalgebras whose carrier has additional structure have been widely considered. Here we are interested in coalgebras the carriers of which are algebras, see eg [6], [19]. Our own interest arises from the following two developments.

First, streams or weighted automata as pioneered by Rutten [15], [16], [17] are mathematically highly interesting examples of coalgebras, despite the fact that the type functor is very simple, eg just

$$HX = A \times X$$

[1] Supported by a Royal Society International Travel Grant

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

in the case of streams. The interesting structure arises from A and in typical examples it will carry the structure of a semi-ring. In this paper, we will bring this structure to the fore by lifting H to the category of modules for a semi-ring, or more generally, to the category of algebras for suitable monads.

Second, in recent work of Kissig and the second author, it turned out that it is of interest to move the trace-semantics of Hasuo et al [7] from the Kleisli-category of a commutative monad to the category of algebras for the monad (for example, this allows to consider wider classes of monads). Again, for trace semantics, semi-ring monads are of special interest.

In Section 2, we show that Barr's theorem [5]—roughly saying that the ω-limit of the final H-sequence is the Cauchy completion of the ω-colimit of the initial H-sequence—extends from Set-coalgebras to $Alg(\mathbf{M})$-algebras for a monad $\mathbf{M}$ on Set. Note that Barr's theorem needs the assumption $H0 \neq 0$, which is not the case for the functor H of stream coalgebras (see above).

We consider the situation of an endofunctor H on Set such that there is a lifting of H to $Alg(\mathbf{M})$. Under some reasonable assumptions we are able to prove that the final H-coalgebra can be obtained as the Cauchy completion of the image of the initial algebra for the lifted functor, under the usual ultrametric inherited from the final sequence. For this, we need to understand better the initial algebra of the lifted functor. This is the purpose of Section 3. For two endofunctors H, T and a monad $\mathbf{M}$ on Set, we call (T, H) an $\mathbf{M}$-commuting pair if there is a natural isomorphism $HM \cong MT$. It follows that if one endofunctor admits an algebra lift $\tilde{H}$, the other endofunctor has a Kleisli lift and the left Kan extension $\bar{T}$ of the Kleisli lift to $Alg(\mathbf{M})$ is isomorphic with the algebra lift $\tilde{H}$, then this provides an example of such a commuting pair. Conversely, one may wonder when two such liftings (when they exist) are isomorphic (in the sense above) for a commuting pair of endofunctors. The result is affirmative in case all functors involved are finitary, with the additional condition that the natural isomorphism $HM \cong MT$ be one of algebras, where the algebra structure on HMX is provided by a distributive law ensuring the lift existence. If this is the case, then one can recover the initial algebra for the lifted endofunctor as the free $\mathbf{M}$-algebra built on the initial T-algebra. Some special cases are considered as examples. A deeper analysis of these will be considered in a forthcoming paper.

2 Final coalgebra for endofunctors lifted to categories of algebras

2.1 Final sequence for Set-endofunctors

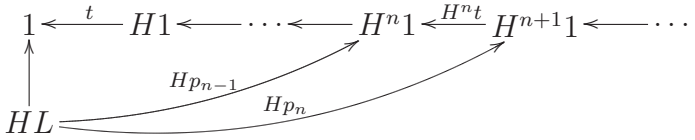
Consider an endofunctor $H : Set \longrightarrow Set$. From the unique arrow $t : H1 \longrightarrow 1$ we may form the sequence

$$1 \xleftarrow{t} H1 \longleftarrow \cdots \longleftarrow H^n 1 \xleftarrow{H^n t} H^{n+1}1 \longleftarrow \cdots \tag{2.1}$$

Denote by L its limit, with $p_n : L \longrightarrow H^n 1$ the corresponding cone. As we work in Set, recall that the limit L can be identified with a subset of the cartesian product $\prod_{n\geq 0} H^n 1$, namely

$$L = \{(x_n)_{n\geq 0} \mid H^n t(x_{n+1}) = x_n\}$$

By applying H to the sequence and to the limit, we get a cone



with $HL \to 1$ the unique map to the singleton set. The limit property leads to a map $\tau : HL \to L$ such that $p_n \circ \tau = Hp_{n-1}$.

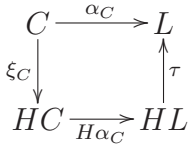
For each H-coalgebra $(C, \xi_C : C \longrightarrow HC)$ it exists a cone $\alpha_n : C \longrightarrow H^n 1$ over the sequence (2.1), built inductively as follows: $\alpha_0 : C \longrightarrow 1$ is the unique map, then if $\alpha_n : C \longrightarrow H^n 1$ is already obtained, construct α_{n+1} as the composite

$$C \xrightarrow{\xi_C} HC \xrightarrow{H\alpha_n} H^{n+1}1 \tag{2.2}$$

Then the unique map $\alpha_C : C \longrightarrow L$ such that

$$p_n \circ \alpha_C = \alpha_n$$

satisfies the following algebra-coalgebra diagram [14]:



On the sequence (2.1), endow each set $H^n 1$ with the discrete topology (so all maps $H^n t$ will be continuous). Then put the initial topology [18] coming from this sequence on L and HL. It follows that τ is continuous. In particular, the topology on L is given by an ultrametric: the distance between

any two points in L is 2^{-n}, where n is the smallest natural number such that $p_n(x) \neq p_n(y)$. The cone $\alpha_n : C \longrightarrow H^n 1$ yields on any coalgebra a pseudo-ultrametric (hence a topology), and the unique map $\alpha_C : C \longrightarrow L$ is continuous with respect to it.

If H is ω^{op}-continuous, it preserves the limit L, hence the isomorphism $\xi = \tau^{-1} : L \simeq HL$ makes L the final H-coalgebra. Moreover, using the above topology, the map ξ is a homeomorphism and verifies

$$Hp_{n-1} \circ \xi = p_n \tag{2.3}$$

2.2 Lifting to Eilenberg-Moore category of algebras for a monad

Let $\mathbf{M} = (M, m : M^2 \longrightarrow M, u : Id \longrightarrow M)$ be a a monad on Set. Denote by $Alg(\mathbf{M})$ the Eilenberg-Moore category of $\mathbf{M}$-algebras and by $F^{\mathbf{M}} \dashv U^{\mathbf{M}} : Alg(\mathbf{M}) \longrightarrow Set$ the adjunction between the free and the forgetful functor. Then $Alg(\mathbf{M})$ has an initial object, namely $(F^{\mathbf{M}}0, m_0 : M^2 0 \longrightarrow M0)$, the free algebra on the empty set, and a terminal object 1, the singleton, with algebra structure given by the unique map $M1 \longrightarrow 1$.

For a Set-endofunctor H, it is well known [9] that liftings of H to $Alg(\mathbf{M})$, i.e. endofunctors $\widetilde{H}$ on $Alg(\mathbf{M})$ such that the diagram

$$\begin{array}{ccc} Alg(\mathbf{M}) & \xrightarrow{\tilde{H}} & Alg(\mathbf{M}) \\ {\scriptstyle U^{\mathbf{M}}}\downarrow & & \downarrow{\scriptstyle U^{\mathbf{M}}} \\ Set & \xrightarrow{H} & Set \end{array} \tag{2.4}$$

commutes, are in one-to-one correspondence with natural transformations $\lambda : MH \longrightarrow HM$ satisfying

$$\begin{array}{ccccccc} H & \xrightarrow{u_H} & MH & M^2H & \xrightarrow{M\lambda} & MHM & \xrightarrow{\lambda_M} & HM^2 \\ & {\scriptstyle Hu}\searrow & \downarrow{\scriptstyle \lambda} & \downarrow{\scriptstyle m_H} & & & & \downarrow{\scriptstyle Hm} \\ & & HM & MH & & \xrightarrow{\lambda} & & HM \end{array} \tag{2.5}$$

Remark 2.1 It is worth noting that the lifting is not unique (as there may be more than one distributive law $\lambda : MH \longrightarrow HM$). For example, take G a group and $HX = MX = G \times X$; consider H as an endofunctor and M as a monad with natural transformations u, m obtained from the group structure. The algebras for this monad are the G-sets. Then it is easy to see that a map $f : G \times G \longrightarrow G \times G$ induces a distributive law $\lambda : MH \longrightarrow HM$ if it satisfies $f(e, x) = (x, e)$ for all $x \in G$, where e stands for the unit of the group, and $f(\mu \times G) = (G \times \mu)(f \times G)(G \times f)$, where we have denoted by μ the group multiplication. Take now $f_1(x, y) = (xy, x)$ and $f_2(x, y) = (xyx^{-1}, x)$; these

maps produce two distributive laws $\lambda_1, \lambda_2 : MH \longrightarrow HM$ which do not give same lifting $\tilde{H}$, as the G-action on HX would be $(x, y, z) \longrightarrow (xy, x \rightharpoonup z)$ for λ_1, respectively $(x, y, z) \longrightarrow (xyx^{-1}, x \rightharpoonup z)$ for λ_2. Here $x, y \in G, z \in X$ and $\rightharpoonup$ stands for the left G-action on X. If the liftings were isomorphic, then the associated categories of coalgebras would also be isomorphic. In particular, notice that H is a comonad (as any set, in particular G, carries a natural comonoid structure) and both maps f_1, f_2 are actually inducing monad-comonad distributive laws λ_1, respectively λ_2. Hence each lifting carries a comonad structure such that the associated categories of coalgebras for the lifted functors are Eilenberg-Moore categories of coalgebras and they should also be isomorphic. But for f_1, a corresponding coalgebra structure is the same as a G-set $(X, \rightharpoonup)$ endowed with a map $\theta : X \longrightarrow G$ such that $\theta(g \rightharpoonup x) = g\theta(x)$, while for the second structure, the compatibility relation yields a crossed G-set, i.e. $\theta(g \rightharpoonup x) = g\theta(x)g^{-1}$.

In particular, for any $\mathbf{M}$-algebra (X, x), HX becomes an algebra with

$$MHX \xrightarrow{\lambda_X} HMX \xrightarrow{Hx} HX$$

and for all algebra maps $(X, x) \longrightarrow (Y, y)$, the corresponding map $HX \longrightarrow HY$ respects the algebra structure. Also, for any H-coalgebra $C \xrightarrow{\xi_C} HC$, MC inherits an H-coalgebra structure by

$$\xi : MC \xrightarrow{M\xi_C} MHC \xrightarrow{\lambda_C} HMC$$

In particular, if the final coalgebra $(L, L \xrightarrow{\xi} HL)$ exists, then there is a unique coalgebra map $\gamma : ML \longrightarrow L$, given by:

$$\begin{array}{ccc} ML & \overset{\gamma}{\dashrightarrow} & L \\ {\scriptstyle M\xi}\downarrow & & \\ MHL & & \downarrow{\scriptstyle \xi} \\ {\scriptstyle \lambda_L}\downarrow & & \\ HML & \overset{H\gamma}{\dashrightarrow} & HL \end{array} \tag{2.6}$$

Then (L, γ) and $(HL, H\gamma\lambda_L)$ are $\mathbf{M}$-algebras and $\xi : (L, \gamma) \longrightarrow (HL, H\gamma\lambda_L)$ becomes an $\mathbf{M}$-algebra map. By the lifting property, $\widetilde{H}(L, \gamma) = (HL, H\gamma\lambda_L)$ and as any $\widetilde{H}$-coalgebra (its underlying set) is the carrier of an H-coalgebra, it follows that $((L, \gamma), \xi)$ is the final $\widetilde{H}$-coalgebra. Hence despite the fact that the lifting might not be unique, the underlying set of the final H-coalgebra is preserved (but with possibly different algebra structure, depending on λ).

Coming back to the final sequence (2.1), note that any term $H^n 1$ has an $\mathbf{M}$-algebra structure, given by:

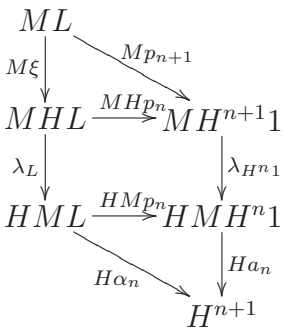
- the obvious unique **M**-algebra structure on 1, $a_0 : M1 \longrightarrow 1$
- given $a_n : MH^n1 \longrightarrow H^n1$, define a_{n+1} as the composite

$$MH^{n+1}1 \xrightarrow{\lambda_{H^n1}} HMH^n1 \xrightarrow{Ha_n} H^{n+1}1 \tag{2.7}$$

Moreover, all maps in the sequence (2.1) are **M**-algebra maps by (2.5). Applying M to the sequence produces a cone from ML to the final sequence. If we assume H is ω^{op}-continuous (hence $\xi : L \simeq HL$ is an isomorphism), we can understand better this cone-construction:

Lemma 2.2 *The cone* $ML \xrightarrow{Mp_n} MH^n1 \xrightarrow{a_n} H^n1$ *coincides with the cone* $\alpha_n : ML \longrightarrow H^n1$ *induced by the* H*-coalgebra structure of* ML *from (2.6).*

Proof. Inductively. For $n = 0$, there is nothing to show as 1 is the terminal object in *Set*. Assume $\alpha_n = a_n Mp_n$, then in the following diagram



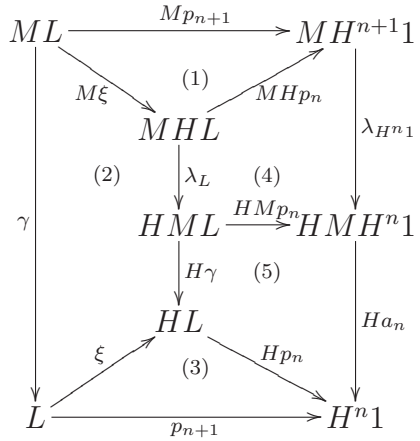
the upper triangle commutes by (2.3), the middle square by naturality of λ and the lower triangle by applying H to the inductive hypothesis. It follows that $\alpha_{n+1} = a_{n+1} Mp_{n+1}$. □

In consequence, the unique coalgebra map $\gamma : ML \longrightarrow L$ constructed in (2.6) is also the unique map $\alpha_{ML} : ML \longrightarrow L$ for the coalgebra ML.

Lemma 2.3 *The projections* $p_n : L \longrightarrow H^n1$ *are* **M***-algebra morphisms, with (2.6) and (2.7) giving the algebra structures of* L*, respectively* H^n1*.*

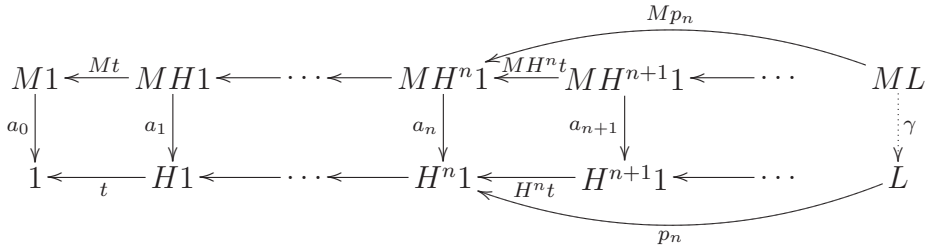
Proof. Again by induction. The first step is trivial. Assume that p_n is an

algebra map: $\pi_n \circ \gamma = a_n \circ Mp_n$; then we have the following diagram



where: (1) commutes by applying M to (2.3); (2) commutes by (2.6); (3) commutes by (2.3); (4) commutes by the naturality of λ and (5) commutes by applying H to the inductive hypothesis. □

Resuming all above, we have the following diagram of **M**-algebras and **M**-algebra morphisms, in which the lower sequence is limiting:



2.3 Topology on the final coalgebra

From now on, we shall assume that H is an ω^{op}-continuous endofunctor which admits a lifting to $Alg(\mathbf{M})$. Remember that all H^n1 were considered with the discrete topology. Endow also all MH^n1 with the discrete topology (intuitively, this corresponds to the fact that operations between algebras with discrete topology are automatically continuous) and ML with the initial topology coming from the cone $Mp_n : ML \longrightarrow MH^n1$ (which is the same as the initial topology from the cone $ML \stackrel{Mp_n}{\longrightarrow} MH^n1 \stackrel{a_n}{\longrightarrow} H^n1$, as a_n are continuous maps bet5ween discrete spaces).

Proposition 2.4 *Under the above assumptions, the final H-coalgebra inherits a structure of a topological* **M***-algebra*[2]*, i.e. L has a* **M***-algebra structure*

[2] Usually the notion of a topological algebra refers to algebra for some finitary, algebraic theory equipped with topologies on the underlying set, so that the algebra operations are

$\gamma : ML \longrightarrow L$ such that γ is continuous with respect to the topologies on L and ML.

Proof. By definition of the initial topology, γ is continuous if and only if all compositions $\gamma \circ p_n$ are continuous. But $\gamma \circ p_n = a_n \circ Mp_n$, a_n are continuous as maps between discrete sets and Mp_n are continuous by the initial topology on ML. □

Notice that this result relies heavily on the construction of the final coalgebra as the limit of the sequence (2.1). Without it, we could not obtain this just by assuming that H has a final coalgebra and H has a lifting to $Alg(\mathbf{M})$, as there is no obvious choice for the topology on ML. Also it can be interpreted as saying that all operations on L are continuous (as they are obtained as limits of operations on discrete algebras).

Remark 2.5 Instead of an ω^{op}-continuous endofunctor, we could use a finitary one. It is known [20] that the final coalgebra exists, but the previous limit yields only a weakly final coalgebra. From this, a supplementary construction gives the final coalgebra. Obviously, the final coalgebra has an $\mathbb{M}$-algebra structure as in (2.6). Following Worrell's construction [20], the terminal sequence would still induce a topology on L, and the easiest way would be to take on ML the initial topology with respect to γ, but this is not the same as the construction pursued here (the topology on ML comes from the terminal sequence).

*2.4 Initial $\tilde{H}$-algebra and final $\tilde{H}$-coalgebra in Alg(**M**)*

If $\tilde{H}$ preserves colimits of ω-sequences, then its initial algebra is easy to build: recall that $Alg(\mathbf{M})$ has an initial object, namely the free algebra on the empty set, $M0$. In order to simplify the notation, we shall identify all algebras $\widetilde{H}^n F^{\mathbf{M}} 0$ with their underlying sets $H^n M0$. Then it is well-known that the initial $\widetilde{H}$-algebra is the colimit in $Alg(\mathbf{M})$ of the chain

$$M0 \stackrel{!}{\longrightarrow} HM0 \stackrel{H!}{\longrightarrow} ... \longrightarrow H^n M0 \stackrel{H^n!}{\longrightarrow} ...$$

where $! : M0 \longrightarrow HM0$ is the unique algebra map. Denote by $i_n : H^n M0 \longrightarrow I$ the colimiting cocone. We do not detail anymore this construction as we did for coalgebras as it will not be used in the sequel. However, we shall need the following (which does not require $\tilde{H}$ to be continuous, just the existence in $Alg(\mathbf{M})$ of limit $\lim H^n 1$ and $\mathrm{colim} H^n M0$): it exists a unique $\mathbf{M}$-algebra

continuous [10]. As Eilenberg-Moore algebras for a *Set*-monad are the same as algebras for (not necessarily) finitary algebraic theories [1], we find that the term topological algebra characterizes the best the present situation.

morphism $f : I \longrightarrow L$ such that

$$\begin{array}{ccc} H^n M0 & \xrightarrow{i_n} & I \\ {\scriptstyle H^n s}\downarrow & & \downarrow{\scriptstyle f} \\ H^n 1 & \xleftarrow{t_n} & L \end{array} \tag{2.8}$$

commutes for all n (see for example [3], Lemma II.5 for a proof), where $s : M0 \longrightarrow 1$ is the unique algebra map from the initial to the final $\mathbf{M}$-algebra. Assume $M0$ not empty, then I will also be not empty, as it comes with a cocone of algebra maps with not empty domains.

We shall generalize in this section the result of Barr [5] from *Set* to $Alg(\mathbf{M})$, for the special case of $Alg(\mathbf{M})$-endofunctors arising as liftings of *Set*-endofunctors. The proofs use similar ideas to the ones in [5] and [3].

We shall assume that there is an algebra map

$$j : 1 \longrightarrow M0 \tag{2.9}$$

As $M0$ is initial, $j \circ s = Id$. By finality of 1 in $Alg(\mathbf{M})$, $s \circ j = Id$, hence we may identify $M0$ and 1 as the zero object in the category of algebras.

Remark 2.6 There is a large class of monads satisfying this condition: the list monad (and the commutative monoid-group-semi-ring monad), the (finite) power-set monad, the maybe monad, the $\Bbbk$-modules monad for a semi-ring $\Bbbk$. For all these, the free algebra with empty generators is built on the singleton. But there are also monads for which the free algebra on the empty set has more than one element, as the exception monad or the families monad, or it is empty, as is the case for the monad $MX = X \times \mathfrak{M}$, for $\mathfrak{M}$ a monoid. It is still under work whether the results of the present paper hold under this weakened assumption.

We have $! : 1 = M0 \longrightarrow HM0 = H1$ and $t \circ ! = Id$ in $Alg(\mathbf{M})$. Hence in the final sequence (2.1) all morphisms are split algebra maps, the colimit is the initial $\widetilde{H}$-algebra and the limit is the final H (and $\widetilde{H}$)-coalgebra:

$$1 \underset{!}{\overset{t}{\leftrightarrows}} H1 \leftrightarrows \ldots \leftrightarrows H^n 1 \underset{H^n !}{\overset{H^n t}{\leftrightarrows}} H^{n+1} 1 \leftrightarrows \ldots \tag{2.10}$$

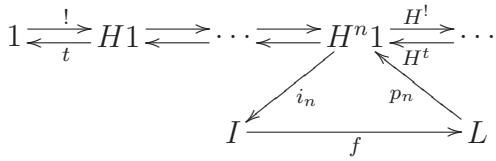
Theorem 2.7 *Let H a Set-endofunctor ω^{op}-continuous, $\mathbf{M}$ a monad on Set such that:*

(i) *H admits a lifting $\tilde{H}$ to $Alg(\mathbf{M})$ which is ω-cocontinuous;*

(ii) *$M0 = 1$ in $Alg(\mathbf{M})$;*

then the final H-coalgebra is the completion of the initial $\tilde{H}$- algebra under a suitable (ultra)metric.

Proof. Consider the following diagram (in $Alg(\mathbf{M})$), where all algebras involved have structure map defined via the distributive law λ.



Put on I the smallest topology such that f is continuous, where L has the structure of a topological algebra from Proposition 2.4. This coincides with the initial topology given by the cone $I \xrightarrow{f} L \xrightarrow{p_n} H^n 1$. Moreover, I becomes a topological algebra and all i_n are continuous algebra maps, if on MI we take the topology induced by the map $Mf : MI \longrightarrow ML$. In particular, Mf is continuous. Denote by $MI \xrightarrow{\zeta} I$ the algebra structure map of I. Then $f \circ \zeta = \gamma \circ Mf$ (remember that f is an algebra map). As L is a topological $\mathbf{M}$-algebra, it follows that $f \circ \zeta$ is continuous, hence ζ is continuous by construction. About i_n: these are by construction algebra maps (as the components of the colimiting cocone in $Alg(\mathbf{M})$) and also continuous, as $H^n 1$ are discrete. The only remaining thing we need to prove is the density of I (more precisely, of Imf) in L. We start by applying Barr's argument to show that L is complete under this ultrametric. First, use that limits in $Alg(\mathbf{M})$ are computed as in Set to conclude that L is Cauchy complete: take a Cauchy sequence $x^{(n)}$ in L in the initial topology (ultrametric) and assume $d(x^{(n)}, x^{(m)}) < 2^{-\min(m,n)}$ for all m, n. This implies $p_n f(x^{(n)}) = p_n f(x^{(m)})$ for all $n < m$. Thus $y = (p_n f(x^{(n)}))_{n \geq 0}$ defines an element of L and $\lim x^{(n)} = y$ with respect to the ultrametric on L. Next, a similar construction to the one in [4] will show us that the image of I under the algebra morphism f is dense in L. For this purpose, consider the additional $\mathbf{M}$-algebra sequence of morphisms $(h_n)_{n \geq 0}$, given by

$$h_n : L \xrightarrow{p_n} H^n 1 = H^n M0 \xrightarrow{H^n !} H^{n+1} M0 \xrightarrow{i_{n+1}} I \xrightarrow{f} L$$

We have $p_{n+1} \circ h_n = H^n ! \circ p_n$. Consider now an element $x \in L$. Then by construction $(y^{(n)} = h_n(x))_{n \geq 0}$ form a sequence of elements lying in the image of f and we shall see that this sequence is convergent to x. Indeed, from $p_{n+1}(y^{(n)}) = H^n ! \circ p_n(x)$ it follows that

$$p_n(y^{(n)}) = H^n \circ t \circ p_{n+1}(y^{(n)}) = H^n \circ t \circ H^n ! \circ p_n(x) = p_n(x)$$

the n-th projection of the n-th term of the sequence $(y^n)_{n \geq 0}$ coinciding with the n-th projection of the element x; hence $d(y^{(n)}, x) < 2^{-n}$ which obviously implies convergence, $\lim y^{(n)} = x$ in L. Therefore the image of I through the canonical colimit$\longrightarrow$limit arrow is dense in L. □

Remark 2.8 (i) If we consider on the initial algebra I the final topology coming from the ω-chain, this is exactly the discrete topology (and metric), since all $H^n 1$ are discrete, hence I would be Cauchy complete and $f : I \to L$ automatically continuous. No interesting information between I and L can be obtained in this situation.

(ii) From (2.9) and (2.8) we have $p_n \circ f \circ i_n = Id$, hence $f \circ i_n$ is a monomorphism. But all morphism in the above sequence are split algebra maps by (2.10), hence all $H^n!$ are mono's. Recall now from [3] that in any locally finitely presentable category,

- the cocone to the colimit of an ω-chain formed by monomorphisms is a monomorphism and
- for every cocone to the chain formed by monomorphisms, the unique map from the colimit is again a monomorphism.

If we assume M finitary, its Eilenberg-Moore category of algebras would be locally finitely presentable. Hence the algebra map f would be mono. But $\mathbf{M}$ is a monad on Set, hence it is regular. It follows that we can identify I with a subalgebra of L. The algebra isomorphism $g : I \simeq Imf$ would also be a homeomorphism, if we take on Imf the induced topology from $L \supseteq Imf$.

(iii) The ω-cocontinuity of $\tilde{H}$ is automatically satisfied if we assume M, H to be finitary. For, the monad being finitary, the forgetful functor $U^{\mathbf{M}}$ would preserve and reflect sifted colimits. But $U^{\mathbf{M}}\tilde{H} = HU^{\mathbf{M}}$, hence $\tilde{H}$ commutes with sifted colimits, in particular with colimits of ω-chains.

Example 2.9 Consider $\Bbbk$ a semi-ring and $\mathbf{M}$ the monad that it induces (as in [12], Section VI.4, Ex. 2, where the ring R is replaced by the semi-ring $\Bbbk$), then $Alg(\mathbf{M})$ is the category of $\Bbbk$-modules and $M0$ is the zero module. Take the Set-endofunctor $HX = \Bbbk \times X^A$, where A is a finite set. Then it is easy to see that a lifting of H exists and it is given by the same formula, where this time the product and the power are computed in the category of modules. The final H-coalgebra is the power $\Bbbk^{A^*}$ (the formal power series in noncommuting A variables), while the initial $\tilde{H}$-algebra is the direct sum of A^* copies of $\Bbbk$ (the polynomial algebra in same variables) (recall that in this case, finite products and coproducts coincide in $Alg(\mathbf{M})$. The approximants of order n in the corresponding ω-sequence are $H^n 1 = \Bbbk^{1+A+\ldots+A^n}$, the polynomials in (non-commuting) A-variables of degree at most n). We shall detail this for the easiest case, where A is the singleton $\{t\}$; the distance between two elements of the final coalgebra $\Bbbk[[t]]$, i.e. between two power series $f(t), g(t)$ in variable t, is given precisely by $2^{-ord(f(t)-g(t))}$, where $ord(f(t) - g(t))$ is the order of the difference $f(t) - g(t)$ (the smallest power of t which occurs with a nonzero coefficient in the difference). Take a Cauchy sequence of polynomials $f_n(t) = a_0^n + a_1^n t + \ldots$, where only finitely many a_j^n are nonzero, for each

$n, j \in \mathbb{N}$. For every $r \geq 0$, there exists an n_r such that for every $n \geq n_r$, we have $ord(f_n(t) - f_{n_r}(t)) = r$; this implies $a_j^n = a_j^{n_r}$ for all $j \leq r$ and $n \geq n_r$. Let $f(t) = a_0^{n_0} + a_1^{n_1} t + \ldots$. One immediately verifies that the power series $f(t)$ is the limit of of the sequence $(f_n(t))_{n \geq 0}$. Hence the final coalgebra $\Bbbk[[t]]$ is indeed the completion of the initial $\tilde{H}$-algebra $\Bbbk[t]$.

3 An application: M-commuting pairs of endofunctors

Consider an endofunctor H and a monad $\mathbf{M}$, both on Set. There are two ways of relating the endofunctor to the monad by a natural transformation, as follows:

- $\lambda : MH \longrightarrow HM$ satisfying (2.5), which is the same as an algebra lift (see 2.4) $\tilde{H} : Alg(\mathbf{M}) \longrightarrow Alg(\mathbf{M})$, $U^{\mathbf{M}} \tilde{H} = H U^{\mathbf{M}}$);
- or $\varsigma : HM \longrightarrow MH$ satisfying

$$\begin{array}{ccccccc} H & \xrightarrow{Hu} & HM & HM^2 & \xrightarrow{\varsigma_M} & MHM & \xrightarrow{M\varsigma} & M^2H \\ & \searrow^{u_H} & \downarrow{\scriptstyle \varsigma} & \downarrow{\scriptstyle Hm} & & & & \downarrow{\scriptstyle m_H} \\ & & MH & HM & & \xrightarrow{\quad\varsigma\quad} & & MH \end{array} \tag{3.1}$$

It is well known that this is equivalent to the existence of a Kleisli lift, i.e. an endofunctor $\hat{H} : Kl(\mathbf{M}) \longrightarrow KL(\mathbf{M})$ such that $\hat{H} F_{\mathbf{M}} = F_{\mathbf{M}} H$, where $F_{\mathbf{M}} : Set \longrightarrow Kl(\mathbf{M})$ is the canonical functor to the Kleisli category of the monad. In this case, we can perform the following additional construction: denote by $\mathcal{I} : Kl(\mathbf{M}) \longrightarrow Alg(\mathbf{M})$ the comparison functor. Take the $Alg(\mathbf{M})$-endofunctor given by the left Kan extension along $\mathcal{I}$ (which exists since every algebra in $Alg(\mathbf{M})$ arises as a coequaliser of free algebras in a canonical way):

$$\bar{H} = Lan_{\mathcal{I}}(\mathcal{I}\hat{H}) \tag{3.2}$$

As the Kleisli category $Kl(\mathbf{M})$ is isomorphic to a full subcategory of $Alg(\mathbf{M})$, this would yield a natural isomorphism $\mathcal{I}\hat{H} \cong \bar{H}\mathcal{I}$. Composing this with the functor $F_{\mathbf{M}}$, we obtain $\bar{H} F^{\mathbf{M}} \cong F^{\mathbf{M}} H$, as in the diagram below:

$$\begin{array}{ccc} Alg(\mathbf{M}) & \overset{\bar{H}}{\dashrightarrow} & Alg(\mathbf{M}) \\ \uparrow{\scriptstyle \mathcal{I}} & & \uparrow{\scriptstyle \mathcal{I}} \\ Kl(\mathbf{M}) & \xrightarrow{\hat{H}} & Kl(\mathbf{M}) \\ \uparrow{\scriptstyle F_{\mathbf{M}}} & & \uparrow{\scriptstyle F_{\mathbf{M}}} \\ Set & \xrightarrow{H} & Set \end{array} \qquad F^{\mathbf{M}} \text{ (left)}, \quad F^{\mathbf{M}} \text{ (right)} \tag{3.3}$$

With the above notations, consider now two *Set*-functors T, H such that exist both a lifting of H and a Kleisli lift of T, and $\tilde{H} \cong \bar{T}$. Then we have

$$\begin{aligned} MT &= U^{\mathbf{M}} F^{\mathbf{M}} T = U^{\mathbf{M}} \bar{T} F^{\mathbf{M}} \\ &\cong U^{\mathbf{M}} \tilde{H} F^{\mathbf{M}} = H U^{\mathbf{M}} F^{\mathbf{M}} = HM \end{aligned}$$

i.e. M acts like a switch (up to isomorphism) between the endofunctors T and H.

Definition 3.1 Let $(\mathbf{M}, m, u)$ be a monad on *Set*. A pair of *Set*-endofunctors (T, H) such that $HM \cong TM$ is called an $\mathbf{M}$-commuting pair.

Notice also that $\tilde{H} \cong \bar{T}$ implies

$$\tilde{H} F^{\mathbf{M}} \cong \bar{T} F^{\mathbf{M}} \cong F^{\mathbf{M}} T$$

hence $HM \cong MT$ is an isomorphism of $\mathbf{M}$-algebras:

$$\begin{array}{ccccc} MHMX & \xrightarrow{\lambda_{MX}} & HM^2X & \xrightarrow{Hm_X} & HMX \\ \downarrow{\scriptstyle \cong Mi_X} & & & & \downarrow{\scriptstyle \cong i_X} \\ M^2TX & & \xrightarrow[m_{TX}]{} & & MTX \end{array} \tag{3.4}$$

Here we have used the distributivity law $\lambda : MH \longrightarrow HM$ to obtain the algebra structure on HMX, for any set X.

Conversely, if (T, H) is an $\mathbf{M}$-commuting pair, one may wonder about their relation with the category of $\mathbf{M}$-algebras. Suppose H has an algebra lifting $\tilde{H}$, T has a Kleisli lift and (3.4) holds; then from $HM \cong MT$ and

$$\begin{aligned} HM &= HU^{\mathbf{M}} F^{\mathbf{M}} = U^{\mathbf{M}} \tilde{H} F^{\mathbf{M}} \\ MT &= U^{\mathbf{M}} F^{\mathbf{M}} T \cong U^{\mathbf{M}} \bar{T} F^{\mathbf{M}} \end{aligned}$$

it follows that $U^{\mathbf{M}} \tilde{H} F^{\mathbf{M}} \cong U^{\mathbf{M}} \bar{T} F^{\mathbf{M}}$, that is, the images of $\tilde{H}$ and $\bar{T}$ on free algebras share (up to bijection) same underlying sets. Taking into account that $HM \cong MT$ is an isomorphism of $\mathbf{M}$-algebras (3.4), we obtain that $\tilde{H} \cong \bar{T}$ on free algebras. Assume now that M, T and H are finitary. Then, by construction, $\bar{T}$ is determined by its action on finitely generated free algebras, and so is $\tilde{H}$ (because it preserves sifted colimits by Remark 2.8(iii)). It follows $\tilde{H} \cong \bar{T}$.

We have obtained thus

Proposition 3.2 *Let H, T two endofunctors on Set and $\mathbf{M}$ a monad on Set. Assume that H has an algebra lift $\tilde{H}$ and T has a Kleisli lift with respect to the monad $\mathbf{M}$. Denote by $\bar{T}$ the corresponding left Kan extension, as in (3.2). Then:*

(i) *If* $\tilde{H} \cong \bar{T}$*, then* (T, H) *form an* $\mathbf{M}$*-commuting pair and* $HM \cong MT$ *is an algebra isomorphism.*

(ii) *Conversely, if* M, H, T *are finitary and* $MT \cong HM$ *as algebras, then* $\tilde{H} \cong \bar{T}$*.*

Example 3.3 Take $TX = 1 + A \times X$, with A finite and $\mathbf{M}$ any monad. Then a Kleisli lifting of T exists, namely for each map $X \longrightarrow^f MY$, take $TX \xrightarrow{f} MTY$ to be the composite

$$TX = 1 + A \times X \stackrel{1+A\times f}{\longrightarrow} 1 + A \times MY \longrightarrow$$
$$1 + M(A \times Y) \longrightarrow M1 + M(A \times Y) \longrightarrow M(1 + A \times Y)$$

where the map $1 + A \times MY \longrightarrow 1 + M(A \times Y)$ is obtained from the canonical strength of the monad, while $1 + M(A \times Y) \longrightarrow M1 + M(A \times Y)$ uses the unit of the monad and $M1 + M(A \times Y) \longrightarrow M(1 + A \times Y)$ comes from the coproduct property. Also, it is easy to see that the extension of T to $\mathbf{M}$-algebras is $\bar{T}X = F^{\mathbf{M}}1 + A \cdot X$, for each algebra X, where this time the coproduct (respectively the copower) is computed in $Alg(\mathbf{M})$. If the category of $\mathbf{M}$-algebras has finite biproducts (as in the case of the monad induced by a semi-ring as in Example 2.9), then $\bar{T}$ is the lifting to $Alg(\mathbf{M})$ of the Set-endofunctor $HX = M1 \times X^A$. Hence (T, H) form a commuting pair.

Corollary 3.4 *Assume the assumptions of Proposition 3.2(ii) hold. If* H *is* ω^{op}*-continuous and* $M0 = 1$ *as* $\mathbf{M}$*-algebras, then the final* H*-coalgebra is the completion of the free* $\mathbf{M}$*-algebra built on the initial* T*-algebra under a suitable metric.*

Proof. Follows from Theorem 2.7, by noticing that the M-image of the initial T-algebra (which exists as T is finitary, hence ω-cocontinuous) is the initial $\bar{T}$-algebra (by construction, $\bar{T}$ is finitary, so ω-cocontinuous), while H and $\tilde{H}$ share same final coalgebra. □

Example 3.5 We come back to the case where the monad is induced by a semi-ring $\Bbbk$, as in Example 2.9. Then the initial T-algebra is A^*, the monoid of all words (including the empty one) built from the alphabet A, i.e. all finite sets of inputs. The free $\mathbf{M}$-algebra built on A^* is the direct sum of A^* copies of $\Bbbk$, that is, the polynomial $\Bbbk$-algebra in uncommuting variables $\Bbbk[A]$ (in the category of $\Bbbk$-semimodules), while the final H-coalgebra is the power $\Bbbk^{A^*}$, the noncommutative power series $\Bbbk$-algebra.

However, the situation described until now in this Section has some deficits:

- For two endofunctors T and H, find the appropriate monad such that (T, H) form a commuting pair. As there is a special bond between algebras of T

and coalgebras of H, it is expected that the general case of any two finitary endofunctors would have no solution.

- If (T, H) is an $\mathbf{M}$-commuting pair, find both distributive laws between H and $\mathbf{M}$, respectively between $\mathbf{M}$ and T. For the second one, there is the following suitable situation: for all commutative monads $\mathbf{M}$ and all analytic functors T, a distributive law $TM \longrightarrow MT$ can be constructed [13]. However, lifting to Eilenberg-Moore category seems to be more problematic, even for simplest cases of polynomial functors, as follows:
  - if H is a constant functor, then the image of H (the set) must be the carrier of an $\mathbf{M}$-algebra;
  - if $HX = A \times X$, and A is the carrier of an algebra, a lift is easily seen to exist, as the forgetful functor $U^{\mathbf{M}}$ preserve products. Conversely, if $\tilde{H}$ is a lifting of H, then there is an algebra structure on A, namely $\tilde{H}(1)$.
  - if $HX = X^n$, a power functor, then the lifting exists as the forgetful functor $U^{\mathbf{M}}$ preserves limits;
  - if $HX = A + X$ or $HX = X + X$, there is no obvious distributive law $\lambda : MH \longrightarrow HM$.
- Assume that $\mathbf{M}$ is a commutative monad, then a tensor product $\otimes$ can be defined on $Alg(\mathbf{M})$ such that the free functor $F^{\mathbf{M}} : (Set, +) \longrightarrow (Alg(\mathbf{M}), \otimes)$ is strong monoidal [8]. If T contains binary products, as $T_1X = A \times X$ or $T_2X = X \times X$, an obvious choice of Kleisli lift would give $\bar{T}_1X = F^{\mathbf{M}}A \otimes X$, respectively $\bar{T}_2X = X \otimes X$, where this time $X \in Alg(\mathbf{M})$. Now recall that the tensor product on $Alg(\mathbf{M})$ is obtained as a reflexive coequalizer, hence if we assume the monad not only commutative but also finitary (as all results in this section rely on the finitariness of M), it follows that the forgetful functor would transform the tensor of any two algebras $(X, x : MX \longrightarrow X), (Y, y : MY \longrightarrow Y)$ into the reflexive coequalizer (computed this time in Set) of the maps

$$M(MX \times MY) \underset{m_{X\times Y} \circ M\varphi_2}{\overset{M(x\times y)}{\rightrightarrows}} M(X \times Y)$$

where $\varphi_2 : MX \times MY \longrightarrow M(X \times Y)$ is the monoidal structure of the monad. In particular, for either one of the endofunctors $T = T_1$, $T = T_2$, a corresponding commuting pair (T, H) exists and can be constructed by the above argument. Moreover such an H is finitary by construction. If H is also ω^{op}-continuous and $M0 = 1$ as algebras, then by Corollary 3.4 the final H-coalgebra is the singleton (as the initial algebras for both T_1, T_2 are the empty set).

4 Conclusions

The general picture behind Barr's theorem is conceptually simpler: if one starts with an arbitrary category $\mathcal{C}$ (with initial object, final object and ω-(co)limits) and a $\mathcal{C}$-endofunctor, then the theorem roughly says that the ω-limit of the terminal sequence is a completion of the ω^{op}-colimit of the initial sequence. Of course an appropriate notion of completion is required; it could be of topological nature (as in [5]), or about ordered structures [3].

Currently we restrict to monads for which the free algebra on the empty set is the singleton. However, dropping this assumption would add more requirements on the endofunctor. Another aspect that we intend to consider is working with accessible Set-endofunctors. About the second part of the paper, the notion of a commuting pair of endofunctors with respect to a monad seems to be new, but a more detailed analysis and examples are necessary in order to better understand this structure. We plan to do this in a future paper.

References

[1] Adámek J., H. Herrlich and G.-E. Strecker, "Abstract and Concrete Categories", John Wiley & Sons, 1990.

[2] Adámek, J. and J. Rosický, *On sifted colimits and generalized varieties*, Theory Appl. Categ. **8** (2001), 33-53.

[3] Adámek, J., *Final coalgebras are ideal completions of initial algebras*, J. Log. Comput. **12** (2002), 217-242.

[4] Adámek, J., *On final coalgebras of continuous functors*, Theor. Comput. Sci. **294** (2003), 3-29.

[5] Barr, M., *Terminal coalgebras in well-founded set theory*, Theor. Comput. Sci. **114** (1993), 299-315.

[6] Corradini A., M. Große-Rhode and R. Heckel, *Structured transition system as lax coalgebras*. In "Coalgebraic methods in computer science CMCS'98", Electr. Notes Theor. Computer Sci. **11** (1998), 22-41.

[7] Hasuo I., B. Jacobs and A. Sokolova, *Generic trace theory*. In "Coalgebraic methods in computer science CMCS'06", Electr. Notes Theor. Computer Sci. **164** (2006), 47-65.

[8] Jacobs, B., *Semantics of weakening and contraction*, Ann. Pure Appl. Logic **69** (1994), 73-106.

[9] Johnstone, P. T., *Adjoint lifting theorems for categories of algebras*, Bull. London Math. Soc. **79** (1975), 294–297.

[10] Johnstone, P. T., "Stone spaces", Cambridge Univ. Press, Cambridge, 1982.

[11] Linton, F.E.J., *Coequalizers in categories of algebras*. In "Seminar on Triples and Categorical Homology Theory", LNM **80**, Springer Verlag, 1969, 75-90.

[12] Mac Lane, S., "Categories for the working mathematician", 2nd Ed., GTM **5**, Springer Verlag, New York, 1998.

[13] Milius, St., Th. Palm and D. Schwencke, *Complete Iterativity for Algebras with Effects*. In "Algebra and Coalgebra in Computer Science", LNCS **5728**, Springer Verlag, 2009, 34-48.

[14] Moss, L., *A Note on Expressive Coalgebraic Logics for Finitary Set Functors*, J. of Logic and Comput. Adv. Acc. 2008.

[15] Rutten J., *Automata and coinduction - an exercise in coalgebra*. In "CONCUR'98", LNCS **1466**, Springer Verlag, 1998, 194-218.

[16] Rutten J., *Behavioural differential equations: A coinductive calculus of streams, automata, and power series*, Report SEN-R0023, CWI, Amsterdam, 2000.

[17] Rutten J., *Coinductive counting with weighted automata*, *J. Autom. Lang. Comb.* **8** (2003), 319-352.

[18] Schaefer H. H., "Topological vector spaces", GTM 3, Springer Verlag, 1971.

[19] Turi D. and G. Plotkin, *Towards a mathematical operational semantics*. In "LICS'97", 1997.

[20] Worrell, J., *On the final sequence of a finitary set functor*, Theor. Comput. Sci. **338** (2005), 184-199.

Families of symmetries as efficient models of resource binding

Vincenzo Ciancia[1]

Institute for Logic, Language and Computation - Amsterdam (NL)

Alexander Kurz[2]

University of Leicester (UK)

Ugo Montanari[3]

Università di Pisa (IT)

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 show that named sets are generalisd by the categorical model of families, that is, free coproduct completions, indexed by symmetries, and explain how *locality of interfaces* gives good computational properties to families. We generalise previous equivalence results by introducing a notion of minimal support in presheaf categories indexed over small categories of monos. 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.

Keywords: Presheaves, Families, Named Sets, History-dependent Automata, Coalgebras, Symmetry Reduction, Partition Refinement

[1] Research supported by the Comunidad de Madrid program *PROMESAS* (S-0505/TIC/0407), and by the VICI grant 639.073.501 of the Netherlands Organization for Scientific Research (NWO)

[2] Research partially supported by EPSRC EP/G041296/1

[3] Research partially supported by the EU FP6-IST IP 16004 project *SENSORIA*

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

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 is how to do this 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* [35] or the π-calculus [36], 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*.

Calculi with resource allocation mechanisms (the so called *nominal calculi*) typically have a notion of bisimulation that does not coincide with the standard one over 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` (see [23,10,9,24,34,33], and the foundational work by Moggi [37]), or to *nominal sets* [25] as done in [38]. Presheaves handle names, and in general resources, as having a *global* meaning across all possible processes. Thus, each freshly generated name must be different from all the previous ones, giving rise to infinite states in the presence of loops. Therefore, the operational semantics of a calculus typically has infinite states even for very simple processes, making it difficult to *compute* the abstract semantics, or to *implement* finite state methods, such as minimisation, equivalence checking or model checking.

Named sets. In the parallel research line of *named sets* [40,41], 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 [41], many formalisms (e.g. Petri nets and process calculi) have been mapped into named sets in a fully abstract way. The most important finding here is that modelling the *symmetry group* of each agent is necessary to have a unique abstract model of the π-calculus, leading to [20,43,21], where a coalgebraic minimsation (partition refinement) algorithm for the π-calculus has been implemented, based on *history-dependent automata*, that is, coalgebras in the category of *named sets*. The importance of modelling symmetries is recognised both in the theory of programming language semantics [45] and in practical applications such as model checking [18]. Due to well known results of group theory (in particular Lagrange's theorem, see e.g. [17], §3.3), finite groups have an efficient representation in terms of generators, which is logarithmic with respect to the size of the group. Moreover, many operations on groups can be computed on the compressed representation [32].

The categorical equivalence between nominal sets, named sets and the

pullback-preserving full subcategory [4] of $\mathtt{Set}^{\mathtt{I}}$, called the *Schanuel topos*, has been established in [27,22]. In [12,13], 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, thus allowing one to reuse the same machinery to represent the semantics of other calculi with names.

Our contribution. An advantage of presheaf categories is the flexibility that can be obtained by varying the index category $\mathtt{C}$, giving rise more complex structures than *pure names* (see e.g. [28], or [3]). This flexibility is lost when using named sets, since the index category is fixed to be $\mathtt{I}$. First, in §2 we introduce *families* as concrete representation of free coproduct completions. Our contribution starts in §3 observing that named sets with symmetries are generalised by the categorical model of *families* over a category of groups of automorphisms and related morphisms, that we call $\mathtt{Sym}(\mathtt{C})$. This model is equivalent in the categorical sense to a full subcategory of $\mathtt{Set}^{\mathtt{C}}$, namely coproducts of *symmetrised representables*, that is, representables quotiented by composition with groups of automorphisms. Presheaves are represented by families as sets of *elements* that have an attached symmetry on their available *local interfaces*.

In a sense, this already generalises the equivalence results of [27,22]. However, the exact characterisation of which presheaves are (isomorphic to) coproducts of symmetrised representables is a difficult problem. Perhaps the most important topic in [25] 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 and the categorical equivalence between the two. In §4 we introduce a general notion of support in presheaf categories. Exploiting this definition, we show that the equivalence result of [27,22] 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. A non-trivial example respecting these conditions is the category $\mathtt{E}$ of finite equivalence relations and injective maps between their underlying sets, used in [3,4] to represent explicit fusions of names in process calculi.

Presheaves and families have a very different nature. We refer to this as *locality of interfaces*. In §5 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

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

systems with interfaces may be related: either assuming a *naming authority* giving a global meaning to each available resource, or relying on locally scoped *links* that connect the different systems.

In §6, 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. We remark that §5 and §6 do not depend on the conditions of §4, but rather they are in the general framework of §3.

Related work. To the best of our knowledge, the study of families for an efficient representation of the semantics of programming languages, and the interpretation of their properties as a theory of *locality of interfaces*, are new and have never been investigated before. Coproducts of symmetrised representables are also interesting as a generalisation of the *analytic functors* of Joyal [30]. This is shown by Adámek and Velebil [2] for the case of *locally presentable* index categories. That research line is different in scope and aim from this work: there, a characterisation of the morphisms between analytic functors (the *regular* natural transformations of [30]) would be desirable, but it is still an open problem. Instead, in §4 we develop an equivalence of categories, characterising *all* the natural trasformations of the subcategory by the means of morphisms of families. Moreover, the conditions of [2] to characterise coproducts of symmetrised representables and ours do not imply each other, and there are examples of categories, relevant for our purposes, that only fall under our conditions (see §4).

2 Background

Here we introduce the basic notions related to the *family* construction $\mathtt{Fam}(\mathtt{C})$, which is a representation of the *free coproduct completion* of $\mathtt{C}$.

Remark 2.1 *(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, that is, limits of small diagrams made up of an arbitrary number of arrows into the same object.

A direct description of the free coproduct completion of a category $\mathtt{C}$ is

obtained by the *family* construction, defined as follows.

Definition 2.2 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 $i \in I$ 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 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. The coproducts in $\mathtt{Fam}(\mathtt{C})$ are *freely* generated, and described as follows.

Definition 2.3 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 Families of symmetries

In this section we introduce a condition on presheaves in $\mathtt{Set}^{\mathtt{C}}$, namely being *coproducts of* symmetrised *representables*. The terminology is borrowed from [2]. In the rest of the paper we will discuss the good computational properties of such a representation, and introduce a representability criterion for presheaves over index categories of monos.

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

First, given a small category $\mathtt{C}$, we define a category of groups of automorphisms, and morphisms between them, that we call $\mathtt{Sym}(\mathtt{C})$.

Definition 3.1 We define the (small) category $\mathtt{Sym}(\mathtt{C})$ of *symmetries over* $\mathtt{C}$:

$$|\mathtt{Sym}(\mathtt{C})| = \coprod_{n \in |C|} \{\Phi \subseteq \mathtt{C}(n,n) \mid \Phi \text{ is a group w.r.t. composition}\}$$
$$\mathtt{Sym}(\mathtt{C})(\Phi_1, \Phi_2) = \{h \circ \Phi_1 \mid h \in \mathtt{C}(dom(\Phi_1), dom(\Phi_2)) \wedge \Phi_2 \circ h \subseteq h \circ \Phi_1\}$$

The identity of each object is $id_\Phi = id_{dom(\Phi)} \circ \Phi = \Phi$; the composition of $f_1 = h_1 \circ \Phi_1$ and $f_2 = h_2 \circ \Phi_2$ is defined as $f_2 \circ f_1 = h_2 \circ h_1 \circ \Phi_1$.

An object of $\mathtt{Sym}(\mathtt{C})$ is just denoted by the group Φ, omitting the index n of the coproduct that is recovered as $dom(\Phi)$, the common domain of all the automorphisms in Φ. Arrows of the category are sets of arrows from $\mathtt{C}$, obtained by composition of a group of isomorphisms with a single arrow. Notice 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, while the composition on the right is composition of sets of arrows, as from Remark 2.1. 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 3.2 *Consider two* $\mathtt{Sym}(\mathtt{C})$ *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 $\mathtt{Sym}(\mathtt{C})$.

Definition 3.3 The embedding $\mathtt{J} : \mathtt{C} \to \mathtt{Sym}(\mathtt{C})$ is defined on objects as $\mathtt{J}(n) = \{id_n\}$ and on arrows as $\mathtt{J}(f) = \{f\}$.

3.2 Coproducts of symmetrised representables as families

Throughout the paper, we let $\mathtt{C}$ denote a small category. We recall that the (covariant) hom functor $\mathtt{C}(n, -) : \mathtt{C} \to \mathtt{Set}$, for n an object of $\mathtt{C}$, acts on each object m as $\mathtt{C}(n, m)$, and on each arrow $f : m_1 \to m_2$ as $\mathtt{C}(n, f)(g : n \to m_1) = f \circ g : n \to m_2$. A *representable* presheaf in $\mathtt{Set}^{\mathtt{C}}$ is a functor which is isomorphic to $\mathtt{C}(n, -)$, for n an object of $\mathtt{C}$.

Definition 3.4 Let Φ be an object of $\mathtt{Sym}(\mathtt{C})$ with domain n. We call a *symmetrised representable* $\mathtt{C}(n, -)_{/\Phi}$ a representable quotiented by the indexed relation $g_1 \equiv_m g_2 \iff \exists \rho \in \Phi. g_1 = g_2 \circ \rho$, for $g_1, g_2 : n \to m$.

The equivalence classes of such a quotient at each index m are conveniently described as the composition of each possible arrow with Φ, that is $(\mathtt{C}(n, -)_{/\Phi})m = \{h \circ \Phi \mid h : n \to m\}$. Hereafter we assume that symmetrised

representables are in this form. Notice that any $f \circ \Phi$ is an arrow of $\mathtt{Sym}(\mathtt{C})$, which gives rise to the representation we propose. For convenience we also state what is the action of symmetrised representables on arrows of $\mathtt{C}$, namely $(\mathtt{C}(n,-)_{/\Phi})f(h \circ \Phi) = f \circ h \circ \Phi$.

Among the presheaves in $\mathtt{Set}^{\mathtt{C}}$, some of them are isomorphic to a *coproduct* of symmetrised representables, giving rise to a full subcategory of $\mathtt{Set}^{\mathtt{C}}$. This subcategory is equivalent to $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$. In the rest of the paper we shall advocate that a representation using families is appealing for computer science applications. First of all, even though the proof of equivalence is easily understood, we make it precise by the means of the following well-known proposition (see [8], Lemma 42), also used in [42], to prove the equivalence between named sets and the Schanuel topos.

Proposition 3.5 *Let* $\mathtt{D}'$ *be a locally small category having small coproducts, and* $\mathtt{D}$ *a small category. A functor* $\mathtt{F} : \mathtt{D} \to \mathtt{D}'$ *can be extended to an equivalence from* $\mathtt{Fam}(\mathtt{D})$ *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* n *in* $|\mathtt{D}|$*, the hom functor* $\mathtt{D}'(\mathtt{F}n, -)$ *preserves coproducts); every object of* $\mathtt{D}'$ *is a coproduct of objects in the image of* $\mathtt{F}$*.*

Here we instantiate the theorem with $\mathtt{D} = \mathtt{Sym}(\mathtt{C})$ and $\mathtt{D}'$ the subcategory of coproducts of symmetrised representables in $\mathtt{Set}^{\mathtt{C}}$. 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. 3.5 is applicable. We now exhibit a functor $\mathtt{F} : \mathtt{Sym}(\mathtt{C})^{op} \to \mathtt{Set}^{\mathtt{C}}$.

Definition 3.6 The functor $\mathtt{F}$ acts on objects as $\mathtt{F}\Phi = \mathtt{C}(dom(\Phi), -)_{/\Phi}$. $\mathtt{F}$ acts on each arrow $h \circ \Phi_1 : \Phi_2 \to \Phi_1$ of $\mathtt{Sym}(\mathtt{C})^{op}$ returning a natural transformation, defined at each index n as $(\mathtt{F}(h \circ \Phi_1))_n(h' \circ \Phi_2) = h' \circ h \circ \Phi_1$.

Next, we show that $\mathtt{F}$ respects the first and second conditions of Prop. 3.5. The third condition is satisfied by construction, when restricting the codomain of $\mathtt{F}$ to symmetrised representables.

Proposition 3.7 $\mathtt{F}$ *is a functor, and in particular an* embedding*, i.e. injective on objects and morphisms. For each object* $\Phi : \mathtt{Sym}(\mathtt{C})$*,* $\mathtt{F}\Phi$ *is indecomposable, that is, the homset functor* $\mathtt{Set}^{\mathtt{C}}(\mathtt{F}\Phi, -)$ *preserves coproducts.*

As $\mathtt{Set}^{\mathtt{C}}$ has coproducts, $\mathtt{F}$ extends to a functor from $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ to $\mathtt{Set}^{\mathtt{C}}$.

Definition 3.8 The functor $\mathtt{Presh} : \mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op}) \to \mathtt{Set}^{\mathtt{C}}$ maps an object $\coprod_{i\in I}\{\Phi_i\}$ into $\coprod_{i\in I} \mathtt{F}\Phi_i$ and an arrow $\langle f, \coprod_{i\in I}\{\mathcal{H}_i^f\}\rangle : \coprod_{i\in I}\{\Phi_i\} \to \coprod_{j\in J}\{\Phi'_j\}$ into the natural transformation $\coprod_{i\in I}(\iota_{f(i)} \circ \mathtt{F}\mathcal{H}_i^f)$, where $\iota_{f(i)}$ denotes the $f(i)^{th}$ injection of the coproduct $\coprod_{j\in J} \mathtt{F}\Phi'_j$.

By definition, each presheaf in the image of $\mathtt{Presh}$ is a coproduct of sym-

metrised representables. The functor is full and faitful, and becomes one direction of a categorical equivalence when its codomain is restricted to its image.

The other direction is given by the functor $\mathtt{K}$ mapping coproducts of symmetrised representables into $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$. The action on objects is rather trivial. Given $P = \coprod_{i\in I} \mathtt{C}(dom(\Phi_i), -)_{/\Phi_i}$, we have $\mathtt{K}P = \coprod_{i\in I}\{\Phi_i\}$. The action on arrows is more interesting: let $Q = \coprod_{j\in J} \mathtt{C}(dom(\Phi_j), -)_{/\Phi_j}$, and $g : P \to Q$ be a natural transformation. We define the morphism between families $\mathtt{K}(g : P \to P') = \langle f, \coprod_{i\in I}\{\mathcal{H}_i^f\}\rangle$. For each $i \in I$, let $g_n(\langle i, id_{dom(\Phi_i)} \circ \Phi_i\rangle) = \langle j, h' \circ \Phi_j\rangle$. Then we let $f(i) = j$ and $\mathcal{H}_i^f = h' \circ \Phi_j$. The function f is well defined by indecomposability of objects in the image of $\mathtt{F}$ (Prop. 3.7), in turn coming from naturality of g.

The action of $\mathtt{K}$ on arrows may be roughly explained by the idea of *local interfaces* in families. This is better understood after having introduced the notion of *orbit* and *representative*, which is done in §4.

4 Pullback-preservation, monos and minimal support

In this section we illustrate a characterisation of the coproducts of symmetrised representables in categories indexed by monos, as functors that preserve all pullbacks. We consider the *finite support* condition in the work by Gabbay and Pitts on nominal syntax [26]: each system has a unique minimal "interface". Preservation of pullbacks means preservation of "intersection of interfaces" in a very general sense, and makes it possible to recover a notion of *support* of an element $x \in Pn$ of a presheaf P over an arbitrary category $\mathtt{C}$ as the *minimal* index n' where an element $x' \in Pn'$ exists, such that $Pfx' = x$ for some arrow f.

The results presented here are similar in spirit to the representation of *analytic functors* as species given by Joyal [30], and therefore to [2], where conditions similar to ours are sketched to identify the coproducts of symmetrised representables. We emphasize that the latter research line aims to characterise and extend Joyal's analytic functors and *regular* natural transformations (the latter is still an open problem), whereas we are interested in *all* natural transformations between two coproducts of symmetrised representables. For this reason, we are able to provide an equivalence of categories. Moreover, the index category in [2] should be locally presentable (or at least should have an initial object, see §3 therein), thus ruling out discrete categories and coproducts of categories (hence our results and [2] are logically independent).

The connection between representability of presheaves as families and pullback preservation has been studied in various works. A well known one is [7]. There, the connection between existence of connected limits, wide pullback

preservation and familial representability is explained. But there the index category of the familial representation is still the same index category $\mathtt{C}$, of the presheaf category, and not a category of symmetries over it. Indeed the latter provides one a bit more structure, which we then use for the symmetry reduction procedure of §6.

The idea of representing pullback-preserving presheaves by families of symmetries comes from Staton [42], 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.

A *wide pullback* is the limit of a cocone of arbitrary cardinality (whereas an ordinary pullback is the limit of a cocone of just two arrows). Notice that in the special case of the Schanuel topos of [42], these diagrams are necessarily finite, and thus wide pullbacks are determined by the binary ones. From now on, we let $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ denote the wide-pullback-preserving full subcategory of $\mathtt{Set}^{\mathtt{C}}$. Our theory can be instantiated under the following conditions.

Criterion 4.1 *We assume that all the arrows of* $\mathtt{C}$ *are monic,* $\mathtt{C}$ *has (small, wide) pullbacks, and 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 cocompleteness. Some examples may clarify the applicability of the characterisation.

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.

Coproducts of categories The coproducts of two non-empty categories certainly does not have an initial object and it is not complete. However, from the programming language semantics perspective, these index categories can be used represent calculi that feature several distinct kinds of agents, each one having a different notion of associated interface.

Finite sets and injections: in this case, the obtained equivalence is that between the Schanuel topos and named sets of [22,27]. The associated categories have been used in a wide range of applications as we already emphasized. The correspondence between families and named sets is made

clear by the categorical definitions given in [44,13]; the category $\mathtt{Symset}$ defined therein is $\mathtt{Sym}(\mathtt{I})$.

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. The *network coordination policies* calculus (NCP) [11], has been developed by the first author et al. in the context of formal methods for service-oriented computing. In the calculus, states are pairs consisting of the network topology, represented as a graph, and a policy, which is a program. Entire fresh sub-topologies can be dynamically allocated along the transitions of the operational semantics. Even though category theory is not used in that work, it seems clear that the semantics can be represented using the standard presheaf approach, with finite graphs and injections as the index category. In NCP, bisimulation is used for the definition of conformance of the specification and the implementation, thus the implementation of an efficient bisimulation checker (taking into account the dynamic allocation capabilities of the framework) is of high relevance. Therefore, the calculus will be an appealing case study for the symmetry reduction algorithm that we sketch in this work.

Fusions: Fusions may be described by an indexing category $\mathtt{E}$ of equivalence relations with monic arrows [3]. This category has pullbacks, falls into the conditions of our framework, and it has a rich structure of objects that is used for fusions (see also [28,34]).

4.1 The symmetric decomposition of a presheaf

We now show that under Crit. 4.1, functors in $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ are isomorphic to coproducts of symmetrised representables, that is objects in the image of the functor $\mathtt{Presh}$. Therefore the full category $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ coincides exactly with the subcategory of coproducts of symmetrised representables.

We pursue our goal emplying Prop. 3.5 again. $\mathtt{F}$ being an embedding, and indecomposability of objects in its image are not affected by the additional hypothesis. However, we must prove that each presheaf in the image of $\mathtt{F}$ is pullback-preserving.

Theorem 4.2 *For each Φ, assuming Crit. 4.1, $\mathtt{F}\Phi$ preserves wide pullbacks.*

The rest of the section is devoted to prove the last required condition of Prop. 3.5, that is, each pullback-preserving presheaf is a coproduct of symmetrised representables. 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 4.3 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 4.4 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 4.5 *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 4.6 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 4.5.

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

Definition 4.8 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 4.9 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. 3.8, completes the categorical equivalence.

Definition 4.10 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* [5] 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 4.11 *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 4.12 *Every presheaf* $\mathtt{G}$ *in* $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ *is isomorphic to* $\mathtt{Presh}(\mathtt{SymDec}(\mathtt{G}))$, *therefore* $\mathtt{Set}^{\mathtt{C}}_{\Diamond}$ *is equivalent to* $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$.

Remark 4.13 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 $Gn = n, Gf = f$ 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 [6] 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

[5] In our case, we should call it the history of *interfaces* along morphisms.
[6] G is different from the final object $\coprod_{i \in 1}\{id_0\}$, having a single element with trivial interface

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 [29], and bisimulation checking [21]).

5 Locality of interfaces: the product construction

In [44], one of the authors extended the equivalence of [27,22] 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 results on the product of named sets presented therein.

Multi-(co)products are a specialisation of the notion of multi-(co)limit, studied in detail by Diers [16]. It is well known (see e.g. [14], 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 5.1 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 5.2 *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 5.3 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 5.4 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 5.5 The *pairing* of $\langle f, \coprod_{k\in K}\{\mathcal{H}_k^f\}\rangle : R \to P$ and $\langle g, \coprod_{k\in K}\{\mathcal{H}_k^g\}\rangle : R \to Q$ is the arrow $\langle h, \coprod_{k\in K}\{\mathcal{H}_k^h\}\rangle$, where $h(k) = \langle f(k), g(k), mcp(\langle \mathcal{H}_k^f, \mathcal{H}_k^g\rangle)\rangle$, and $\mathcal{H}_k^h = u_{\langle \mathcal{H}_k^f, \mathcal{H}_k^g\rangle}$.

Theorem 5.6 *The product, projections and pairing given above identify up to isomorphism the binary product in* $\mathtt{Fam}(\mathtt{C}^{op})$.

In the above definition, $mcp(\langle \mathcal{H}_k^f, \mathcal{H}_k^g\rangle)$ and $u_{\langle \mathcal{H}_k^f, \mathcal{H}_k^g\rangle}$ come from Def. 5.1. 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.

6 Symmetry reduction by final semantics

The presheaf approach to operational semantics roughly consists in defining a presheaf P of *terms*, that is, the initial algebra of some endofunctor over a presheaf category, 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 [31] 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 [15] and subsequent works, or the more recent [19]). This is typically done exploiting equations on the syntax of calculi, or by adding symmetry information "by hand" to models. Our approach is very different: 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 [18].

6.1 Symmetry reduction

Remark 6.1 Equivalences extend to categories of coalgebras of suitable "equivalent" endofunctors. In particular, each endofunctor $\mathtt{T}'$ over the full subcategory of coproducts of symmetrised representables in $\mathtt{Set}^{\mathtt{C}}$ 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 $\mathtt{T} = \mathtt{SymDec} \circ \mathtt{T}' \circ \mathtt{Presh}$.

We assume in the following such a pair of equivalent endofunctors $\mathtt{T}'$ and $\mathtt{T}$. Even if for the scope of this work the given definition of $\mathtt{T}$ is sufficent, 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 §5 is isomorphic to the one that we just mentioned, but not the same. This topic has been studied in detail in [44].

We now observe that each natural transformation between coproducts of symmetrised representables 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 6.2 Let R_n^f 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 $\mathcal{G}_x^h = \{\rho : n \to n \mid \mathtt{G}\rho x R_n^f x\}$ the symmetry on x induced by f.

Proposition 6.3 *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 $(\mathtt{F}\Phi_i \rho(h \circ \Phi_i))R_n^f(h \circ \Phi_i)$ if and only if $\rho \circ h \circ \mathcal{H}_i^g = h \circ \mathcal{H}_i^g$.*

Observe that $\rho \circ h \circ \mathcal{H}_i^g = h \circ \mathcal{H}_i^g$ implies that, for each h' in $h \circ \mathcal{H}_i^g$, 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 $h \circ \mathcal{H}_i^g$ 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 $h' \in h \circ \mathcal{H}_i^g$, 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 $h \circ \mathcal{H}_i^g$ 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 $h \circ \mathcal{H}_i^g$. 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.

6.2 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 $\mathtt{Set}^{\mathtt{C}}$, $s : P \to \mathtt{T}'P$ for P representing the syntax. As we know (see Rem. 6.1), if P is a coproduct of symmetrised representables, 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 $\mathtt{T}'$.

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 6.4 Coalgebraic partition refinement in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$ is an iterative algorithm using three variables, f, h and z, denoting arrows in $\mathtt{Fam}(\mathtt{Sym}(\mathtt{C})^{op})$.

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.

Iteration 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 **Iteration step**(f', z', w').

Correctness of the algorithm is well known by the theory of coalgebras (see e.g. [46]). An intuition can be given as follows. At the n^{th} 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 n^{th} 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.

Convergence of the algorithm is equivalent to deciding the semantics of a program, therefore it can not be guaranteed *a priori* for all calculi. For Turing-equivalent languages, the algorithm converges on an undecidable subset of all the possible programs. In labelled transition systems, one gets convergence if the set of states reachable from a given set of initial states is *finite*. When using coalgebras over presheaves, even trivial programs have infinite states, but finiteness of the elements of the corresponding family is enough to guarantee convergence. This leads to a more refined notion of finiteness for presheaves.

Static constraints may be used (e.g. the *finite-control* π-calculus agents of [21]) to identify a subset of the convergent instantiations of the algorithm.

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 [13] such that the algorithm above is the partition refinement procedure for the π-calculus of [39,21].

Computing the semantics

Two basic assumptions are needed. First, objects and arrows of $\mathtt{C}$ should be "finite", in the sense that they can be represented as data structures. Then, f should be computable in each step of the algorithm. Without these assumptions, the algorithm can not be implemented. Indeed, the cases studied in the literature on presheaves for process calculi fall under these hypotheses.

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. 5.6, 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. 5.2, 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 6.5 *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. 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.

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 [44] such a "finitistic" representation is given for the *early semantics* of the π-calculus, which is defined as an infinitary transition system, due to the input transitions.

Under the above restrictions, one has to check if $z = \langle f_z, \coprod_{i\in Im(\mathtt{T}h\circ f)}\{\mathcal{H}_i^{f_z}\}\rangle$ 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 [44] 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.

6.3 Garbage collection

We consider the representation using families appealing because it may allow one to implement iteration along the terminal sequence, starting from a coalgebra defining 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 [24], 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 [12]). Using families, on the other hand, 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 "sub-interface" of each source state, which is preserved in the destination state, thus discarding unused resources.

7 Concluding remarks

We have introduced 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 using *families*. This representation removes the redundant information coming from the notion of interfaces being *global* rather than *local*.

First of all, a complete example of application should be developed. The field of presheaf semantics for process calculi is still a relatively new research field, and there is not so much literature on calculi different from the π-calculus. However, by providing a representation theory, we prepare the grounds on which to build up new applications. An interesting case study is [4], since the presheaf category employed there respects the conditions of §4.

Applications are of great interest in the area of *service-oriented computing*, where resource allocation in the presence of *network topologies* [11], or constraints [6] 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 [32]. Finally, similar consideration apply to model checking. The study of a Stone-type duality for coalgebras over families in a similar fashion to [5], 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 presheaves, 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. *LNCS - Festschrift for Ugo Montanari*, 5065, 2008.

[4] F. Bonchi, M. Buscemi, V. Ciancia, and F. Gadducci. A presheaf environment for the calculus of explicit fusions. *Submitted*, 2009.

[5] M. M. Bonsangue and A. Kurz. Pi-calculus in logical form. In *LICS*, pages 303–312. IEEE Computer Society, 2007.

[6] 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.

[7] A. Carboni and P. Johnstone. Connected limits, familial representability and the artin glueing. *Mathematical Structures in Computer Science*, 5, 1995.

[8] A. Carboni and E. Vitale. Regular and exact completions. *Journal of Pure and Applied Algebra*, 125(1-3):79 – 116, 1998.

[9] G. L. Cattani and P. Sewell. Models for name-passing processes: Interleaving and causal. In *LICS*, pages 322–332, 2000.

[10] G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the π-calculus. In *Category Theory and Computer Science*, pages 106–126, 1997.

[11] V. Ciancia, G. L. Ferrari, R. Guanciale, and D. Strollo. Event based choreography. *Science of Computer Programming*, To appear.

[12] V. Ciancia and U. Montanari. A name abstraction functor for named sets. *Electr. Notes Theor. Comput. Sci.*, 203(5):49–70, 2008.

[13] V. Ciancia and U. Montanari. Symmetries, local names and dynamic (de)-allocation of names. *Information and Computation*, 2009. To appear.

[14] C. Cirstea. Semantic constructions for the specification of objects. *Theor. Comput. Sci.*, 260(1-2):3–25, 2001.

[15] E. M. Clarke, E. A. Emerson, S. Jha, and A. P. Sistla. Symmetry reductions in model checking. In *Computer Aided Verification, 10th International Conference*, volume 1427 of *LNCS*, pages 147–158, 1998.

[16] Y. Diers. Familles universelles de morphismes. *Ann. Soc. Sci. Bruxelles*, 93:175–195, 1979.

[17] J. D. Dixon and B. Mortimer. *Permutation Groups*, volume Permutation Groups of *Graduate Texts in Mathematics*. Springer, 2006.

[18] E. A. Emerson and A. P. Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1/2):105–131, 1996.

[19] 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.

[20] 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.

[21] 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.

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

[23] M. P. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the pi-calculus (extended abstract). In *LICS*, pages 43–54, 1996.

[24] M. P. Fiore and D. Turi. Semantics of name and value passing. In *LICS*, pages 93–104, 2001.

[25] M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *LICS*, pages 214–224, 1999.

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

[27] 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.

[28] N. Ghani, K. Yemane, and B. Victor. Relationally staged computations in calculi of mobile processes. *Electr. Notes Theor. Comput. Sci.*, 106:105–120, 2004.

[29] 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.

[30] A. Joyal. Foncteurs analytiques et espces de structures. In *Combinatoire Ènumérative*, volume 1234 of *Springer Lecture Notes in Mathematics*. Springer Verlag, 1985.

[31] A. Kurz. *Logics for Coalgebras and Applications for Computer Science*. PhD thesis, Ludwig-Maximilians-Universitat Munchen, 2000.

[32] E. M. Luks. Permutation Groups and Polynomial Time Computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.

[33] M. Miculan. A categorical model of the fusion calculus. *Electr. Notes Theor. Comput. Sci.*, 218:275–293, 2008.

[34] 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.

[35] R. Milner. A calculus of communicating systems. *Lecture Notes in Computer Science*, 92, 1980.

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

[37] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[38] U. Montanari and M. Pistore. pi-calculus, structured coalgebras, and minimal hd-automata. In *MFCS*, volume 1893 of *LNCS*, pages 569–578, 2000.

[39] U. Montanari and M. Pistore. Structured coalgebras and minimal hd-automata for the pi-calculus. *Theoretical Computer Science*, 340:539–576, 2005.

[40] U. Montanari, M. Pistore, and D. Yankelevich. Efficient minimization up to location equivalence. In *ESOP*, pages 265–279, 1996.

[41] 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.

[42] S. Staton. Name-passing process calculi: operational models and structural operational semantics. Technical Report UCAM-CL-TR-688, University of Cambridge, Computer Laboratory, 2007.

[43] E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003. TD-8/03.

[44] Vincenzo Ciancia. *Accessible Functors and Final Coalgebras for Named Sets*. PhD thesis, University of Pisa, 2008.

[45] G. Winskel. Symmetry and concurrency. In *CALCO*, pages 40–64, 2007.

[46] J. Worrell. Terminal sequences for accessible endofunctors. *Electr. Notes Theor. Comput. Sci.*, 19, 1999.

Generic Infinite Traces and Path-Based Coalgebraic Temporal Logics

Corina Cîrstea[1]

School of Electronics and Computer Science
University of Southampton, UK

Abstract

This paper gives a general coalgebraic account of the notions of *possibly infinite trace* and *possibly infinite execution* in state-based, dynamical systems, by extending the generic theory of finite traces and executions developed by Hasuo and coauthors [8]. The systems we consider are modelled as coalgebras of endofunctors obtained as the composition of a computational type (e.g. nondeterministic or stochastic) with a general transition type. This generalises existing work by Jacobs [10] that only accounts for a nondeterministic computational type. We subsequently introduce path-based temporal (including fixpoint) logics for coalgebras of such endofunctors, whose semantics is based upon the notion of possibly infinite execution. Our approach instantiates to both nondeterministic and stochastic computations, yielding, in particular, path-based fixpoint logics in the style of CTL* for nondeterministic systems, as well as generalisations of the logic PCTL for probabilistic systems.

Keywords: coalgebra, trace semantics, temporal logic, nondeterminism, probability

1 Introduction

Path-based temporal logics are commonly used as specification logics, particularly in the context of automatic verification. Instances of such logics include the logic CTL* with its fragments CTL and LTL for transition systems [3], and the logic PCTL for probabilistic transition systems [7]. In spite of the similarities shared by these logics, no general, unified account of path-based temporal logics exists.

Coalgebras are by now recognised as a truly general model of dynamical systems, instances of which subsume transition systems, their probabilistic counterparts, and many other interesting state-based models [14]. Moreover,

[1] Email: cc2@ecs.soton.ac.uk

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

the modal logics associated with coalgebraic models [13,1,2] are natural logics for specifying system behaviour, that also instantiate to familiar logics in particular cases. These logics can be classified into one-step modal logics, wherein the semantics of modal operators depends solely on the one-step behaviour of system states (as considered e.g. in [13,1]), and extensions of such logics with (e.g. fixpoint) operators whose interpretation depends on the long-term, possibly infinite behaviour of system states [2]. While (some of) the logics in the second category are able to express application-relevant temporal properties of states, their syntax does not directly refer to the computation paths from particular states, as is the case for logics such as CTL* and PCTL. Indeed, there is still no general, coalgebraic account of the notion of (infinite) computation path, as used in the semantics of CTL* and PCTL. Worse still, in the case of probabilistic transition systems, adding standard fixpoints to the corresponding one-step modal language (as considered in [12,1]) is not very useful, as it does not appear to allow the specification of properties such as: "the likelihood of some state property p holding eventually is greater than some $q \in [0, 1]$".

In what follows, we give a general account of the notion of computation path, and of path-based temporal logics such as CTL* and PCTL. Following [10,8], we model systems as coalgebras of a signature functor obtained as the composition of a computational type T (called *branching type* in [8]) with a transition type F, and require that T distributes over F in a suitable way. As examples, we consider nondeterministic and probabilistic systems, with the non-empty powerset functor $\mathcal{P}^+ : \mathsf{Set} \to \mathsf{Set}$ on the category of sets and respectively the probability measure functor $\mathcal{G}_1 : \mathsf{Meas} \to \mathsf{Meas}$ on the category of measurable spaces describing the computational types needed to recover the usual notions of computation path for such systems. While the transition type describes the type of individual transitions (typically linear) and determines the notion of computation path, the computational type describes how the transitions from particular states are structured (e.g. using sets, or probability distributions). The distributivity of T over F then allows computation paths from individual states to be similarly structured.

Our approach to defining infinite computation paths builds on earlier work by Jacobs [10] where *infinite trace maps* were defined for coalgebras of type $\mathcal{P} \circ F$, with $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ the powerset functor and $F : \mathsf{Set} \to \mathsf{Set}$ a polynomial functor. We generalise this to arbitrary computational types T (subject to some additional constraints), thereby obtaining notions of *possibly infinite trace* and *possibly infinite execution* of a state in a $T \circ F$-coalgebra, that are parametric in T and F. We subsequently introduce path-based temporal (including fixpoint) logics for coalgebras of endofunctors of type $T \circ F$, whose semantics is defined in terms of the possibly infinite executions from a particular state. By instantiating our approach, we recover known temporal logics

and obtain new variants of known logics. Specifically, taking T to be the non-empty powerset monad $\mathcal{P}^+$ and $F = \mathsf{Id}$ sheds new light on the logic CTL* [3], which we recover as a fragment of a path-based fixpoint logic for $\mathcal{P}^+ \circ \mathsf{Id}$. Varying F to $A \times \mathsf{Id}$ with A a set of labels yields an interesting variant of CTL* interpreted over labelled transition systems. On the other hand, taking $T = \mathcal{G}_1$ and $F = \mathsf{Id}$ allows us to recover the logic PCTL [7] as an instance of a generic temporal logic with Until operators.

The paper is structured as follows. The remainder of this section gives a brief overview of the logics CTL* and PCTL, our main examples. Section 2 recalls some basic definitions and results required later and some details of the generic theory of finite traces [8]. Section 3 defines infinite traces and executions and studies their properties. Section 4 uses infinite executions to define general path-based coalgebraic logics, including fixpoint logics and temporal logics with Until operators. A summary of the results and an outline of future work are given in Section 5.

Transition systems and the logic CTL*

The semantics of CTL* [5] is based on the notion of computation path. Given a transition system with set of states S and accessibility relation $R \subseteq S \times S$, a *computation path* from a state s_0 is an infinite sequence of states $s_0 s_1 \ldots$ such that $s_i R s_{i+1}$ for $i \in \omega$. The syntax of CTL* consists of *path formulas*, formalising properties of computation paths and employing operators such as $\mathbf{X}$ (in the *next* state along the path), $\mathbf{F}$ (at some *future* state along the path), $\mathbf{G}$ (*globally* along the path) and $\mathbf{U}$ (*until* operator), and *state formulas*, formalising properties of states and employing operators ($\mathbf{A}$ and $\mathbf{E}$) that quantify (universally, respectively existentially) over the computation paths from a particular state. Every state formula is also a path formula, with the latter requiring that the first state of a path satisfies the given state formula. For example, the property "along every path, the system will eventually reach a success state" is formalised as $\mathbf{A}\,\mathbf{F}\,\mathit{success}$, or equivalently as $\mathbf{A}(\mathsf{tt}\,\mathbf{U}\,\mathit{success})$, where *success* denotes an atomic proposition. The assumption one typically makes of the transition system of interest is that each state s has at least one outgoing transition. (For states where this is not the case, self-loops are added to the original transition system.) This allows one to focus only on the *infinite* computation paths.

Probabilistic transition systems and the logic PCTL

In the probabilistic transition system model, the state transitions are governed by a probability distribution on the target states – this assigns a probability value to each outgoing transition from a particular state, with the values for transitions from the same state summing up to 1. The logic PCTL [7] for

probabilistic transition systems is similar in spirit to CTL*: its syntax consists of path and state formulas, with similar operators ($\mathbf{X}$ and $\mathbf{U}$) for the path formulas, and its semantics is based on the same notion of computation path; the main difference is that, instead of stating that a path formula holds in all/some of the paths from a particular state, the basic state formulas of PCTL, of the form $[\varphi]_{\sim p}$ with φ a path formula and $\sim \in \{<, \leq, >, \geq\}$, refer to the likelihood of φ holding along the paths from a particular state. For example, $[\mathsf{tt}\,\mathbf{U}\,\mathit{success}]_{\geq 1}$ states that the likelihood of eventually reaching a success state is 1. To interpret state formulas, one computes probability measures over the computation paths from each state of a given model.

The previous examples suggest that a general account of computation paths (to be referred to as *infinite executions* in what follows) should first define the shape of a *potential* infinite execution (in the above cases, any infinite sequence of states), and then provide a suitable structure on the *actual* infinite executions from each state of a particular model (e.g. a *set* of computation paths, or a *probability measure* over computation paths). The former should be sufficient to allow an interpretation of path formulas (of a generic path-based logic still to be defined), whereas the latter should support an interpretation of state formulas (of the same logic).

Acknowledgement

Thanks are due to Bart Jacobs for pointing out the notion of affine monad and its relevance to this work, and to the anonymous referees of an earlier version of this paper for their useful comments and suggestions.

2 Preliminaries

We recall that a *measurable space* is given by a pair (X, Σ_X) with X a set and Σ_X a σ-algebra of (*measurable*) subsets of X, whereas a *measurable map* between (X, Σ_X) and (Y, Σ_Y) is given by a function $f : X \to Y$ with the property that $f^{-1}(V) \in \Sigma_X$ for each $V \in \Sigma_Y$. We write Meas for the category of measurable spaces and measurable maps. A measurable space (X, Σ_X) is called *discrete* if $\Sigma_X = \mathcal{P}X$. A *subprobability measure* on a measurable space (X, Σ_X) is then a function $\mu : \Sigma_X \to [0, 1]$ such that $\mu(\emptyset) = 0$ and $\mu(\bigcup_{i\in\omega} X_i) = \sum_i \mu(X_i)$ for countable families $(X_i)_{i\in\omega}$ of pairwise disjoint measurable subsets of X. Thus, $\mu(X) \leq 1$ for any subprobability measure μ on (X, Σ_X). If $\mu(X) = 1$, then μ is called a *probability measure*. Given a measurable space (X, Σ_X) and $x \in X$, the *Dirac probability measure* δ_x is defined by $\delta_x(U) = 1$ iff $x \in U$ and $\delta_x(U) = 0$ otherwise.

We write $\mathcal{G} : \mathsf{Meas} \to \mathsf{Meas}$ for the *subprobability measure functor* [6], sending a measurable space (X, Σ_X) to the set $\mathcal{M}(X, \Sigma_X)$ of subprobability measures on (X, Σ_X), equipped with the σ-algebra generated by the sets $\{\mu \mid \mu(U) \geq q\}$ with $U \in \Sigma_X$ and $q \in [0, 1]$. A related functor, considered in [8], is the *subprobability distribution functor* $\mathcal{S} : \mathsf{Set} \to \mathsf{Set}$, sending a set X to the set of *subprobability distributions over* X, i.e. functions $\mu : X \to [0, 1]$ with $\sum_{x \in X} \mu(x) \leq 1$ [2].

Given a functor $F : \mathsf{C} \to \mathsf{C}$, an *$F$-coalgebra* is given by a pair (X, γ) with X a C-object and $\gamma : X \to FX$ a C-arrow. As previously mentioned, we work in the setting of coalgebras of endofunctors obtained as the composition of a computational type with a transition type. The computational type is specified by a monad T on a category C, whereas the transition type is captured by an endofunctor F on C. As in [8], a crucial assumption is the existence of a distributive law $\lambda : F \circ T \Rightarrow T \circ F$ of T over F. Such a distributive law must be compatible with the monad structure, i.e. $\lambda \circ F\eta = \eta_F$ and $\lambda \circ F\mu = \mu_F \circ T\lambda \circ \lambda_T$, where $\eta : \mathsf{Id} \Rightarrow T$ and $\mu : T^2 \Rightarrow T$ denote the unit and multiplication of the monad T.

As examples of computational types, we consider (variants of):

- the powerset monad $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$, modelling nondeterministic computations, with unit given by singletons and multiplication given by unions,
- the subprobability measure monad $\mathcal{G} : \mathsf{Meas} \to \mathsf{Meas}$, modelling probabilistic computations, with unit given by the Dirac measures and multiplication given by integration (see [6] for details).

Both of the above monads are *strong* and *commutative*, i.e. they come equipped with a *strength map* $st_{X,Y} : X \times TY \to T(X \times Y)$ as well as a *double strength map* $dst_{X,Y} : TX \times TY \to T(X \times Y)$, for each choice of C-objects X, Y [3]:

- the powerset monad has strength given by $st_{X,Y}(x, V) = \{x\} \times V$ and double strength given by $dst_{X,Y}(U, V) = U \times V$, for $x \in X$, $U \in \mathcal{P}X$ and $V \in \mathcal{P}Y$,
- the subprobability measure monad has strength given by $st_{(X,\Sigma_X),(Y,\Sigma_Y)}(x, \nu)(U, V) = \nu(V)$ iff $x \in U$ and $st_{(X,\Sigma_X),(Y,\Sigma_Y)}(x, \nu)(U, V) = 0$ otherwise, and double strength given by $dst_{(X,\Sigma_X),(Y,\Sigma_Y)}(\mu, \nu)(U, V) = \mu(U) \cdot \nu(V)$, for $x \in X$, $\mu \in \mathcal{M}(X, \Sigma_X)$, $\nu \in \mathcal{M}(Y, \Sigma_Y)$, $U \in \Sigma_X$, $V \in \Sigma_Y$.

It is shown in [8] that any commutative monad on Set has a canonical distributive law over any *shapely polynomial functor* (i.e. a functor built from identity and constant functors using finite products and arbitrary coproducts). This provides examples of distributive laws of the powerset monad over shapely polynomial functors. Moreover, the construction of the canonical dis-

[2] Thus, a subprobability distribution can take non-zero values on at most countably-many elements of X.

[3] Moreover, these are natural in X and Y.

tributive law (by induction on the structure of the shapely functor) generalises straightforwardly to any category with products and coproducts, thereby also providing examples of distributive laws of the subprobability measure monad over shapely polynomial functors on Meas.

As in [8], the *Kleisli category* of a monad (T, η, μ) on a category C will play an important rôle when defining the notions of infinite trace and infinite execution for systems whose computational type is given by T. This category, denoted $\mathsf{Kl}(T)$, has the same objects as C, and C-arrows $f : X \to TY$ as arrows from X to Y. The composition of two $\mathsf{Kl}(T)$-arrows $f : X \to Y$ and $g : Y \to Z$ is given by the C-arrow $\mu_Z \circ Tg \circ f$. We let $K : \mathsf{Kl}(T) \to \mathsf{C}$ denote the functor defined by:

- $K(X) = TX$,
- $K(f) = \mu_Y \circ Tf$ for $f : X \to Y$ in $\mathsf{Kl}(T)$,

and write $J : \mathsf{C} \to \mathsf{Kl}(T)$ for its left adjoint, defined by:

- $J(X) = X$,
- $J(f) = Tf \circ \eta_X = \eta_Y \circ f$ for $f : X \to Y$ in C.

Later we will make use of the following property of the functor J:

Lemma 2.1 *If the functor $T : \mathsf{C} \to \mathsf{C}$ (weakly) preserves the limit $(Z, (\pi_i)_{i\in\omega})$ of an ω^{op}-chain $(f_i)_{i\in\omega}$, then so does $J : \mathsf{C} \to \mathsf{Kl}(T)$.*

Proof. Assume first that T weakly preserves the limit $(Z, (\pi_i : Z \to Z_i)_{i\in\omega})$ of $(f_i : Z_{i+1} \to Z_i)_{i\in\omega}$. To show that $(JZ, (J\pi_i : JZ \to JZ_i)_{i\in\omega})$ is a weakly limiting cone for $(Jf_i : JZ_{i+1} \to JZ_i)_{i\in\omega}$ in $\mathsf{Kl}(T)$, let $(X, (\delta_i : X \to JZ_i)_{i\in\omega})$ denote an arbitrary cone for $(Jf_i)_{i\in\omega}$ in $\mathsf{Kl}(T)$. Hence, in C, $\mu_{Z_i} \circ T\eta_{Z_i} \circ Tf_i \circ \delta_{i+1} = \delta_i$, that is, $Tf_i \circ \delta_{i+1} = \delta_i$ for all $i \in \omega$. This makes $(\delta_i)_{i\in\omega}$ a cone over $(Tf_i)_{i\in\omega}$ in C, and the weak limiting property of $(TZ, (T\pi_i)_{i\in\omega})$ in C now yields a mediating map $m : X \to TZ$ such that $T\pi_i \circ m = \delta_i$ in C for all $i \in \omega$. This is equivalent to $\mu_{Z_i} \circ T\eta_{Z_i} \circ T\pi_i \circ m = \delta_i$ in C for $i \in \omega$, that is, $J\pi_i \circ m = \delta_i$ in $\mathsf{Kl}(T)$ for $i \in \omega$. The proof of the stronger statement, in the case when T preserves the limit of $(f_i)_{i\in\omega}$, is similar.

As mentioned above, we assume the existence of a distributive law λ of the monad T over the endofunctor F. It is known (see e.g. [8]) that such distributive laws $\lambda : F \circ T \Rightarrow T \circ F$ are in one-to-one correspondence with liftings of the functor $F : \mathsf{C} \to \mathsf{C}$ to $\mathsf{Kl}(T)$. In particular, the lifting $\overline{F} : \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ induced by a distributive law $\lambda : F \circ T \Rightarrow T \circ F$ is defined by:

- $\overline{F}A = FA$,
- $\overline{F}f = \lambda_B \circ Ff$ for $f : A \to B$ in $\mathsf{Kl}(T)$.

The following property of this lifting will be used later:

Lemma 2.2 *The lifting $\overline{F}$ satisfies $\overline{F} \circ J = J \circ F$.*

Proof. For $f : X \to Y$ in C, the C-arrows that define the Kleisli maps $\overline{F}Jf$ and JFf are $\lambda_Y \circ F\eta_Y \circ Ff$ and respectively $\eta_{FY} \circ Ff$. By the compatibility of the distributive law λ with the monad structure, these coincide.

In what follows we also assume that $\mathsf{Kl}(T)$ is *DCpo-enriched*, that is, each homset $\mathsf{Kl}(T)(X, Y)$ is a partial order, with directed collections of maps $(f_i : X \to Y)_{i\in I}$ admitting a join $\bigsqcup_{i\in I} f_i : X \to Y$, and with composition of arrows preserving directed joins: $g \circ (\bigsqcup_{i\in I} f_i) = \bigsqcup_{i\in I}(g \circ f_i)$ and $(\bigsqcup_{i\in I} f_i) \circ h = \bigsqcup_{i\in I}(f_i \circ h)$. We note that the Kleisli categories of the monads $\mathcal{P}$ and $\mathcal{G}$ are *DCpo*-enriched, with the order on $\mathsf{Kl}(\mathcal{P})(X, Y)$ being defined pointwise via the inclusion order on $\mathcal{P}(Y)$, and the order on $\mathsf{Kl}(\mathcal{G})((X, \Sigma_X), (Y, \Sigma_Y))$ being defined pointwise from the dcpo $\leq_Y$ on $\mathcal{G}(Y, \Sigma_Y)$ given by $\mu \leq_Y \nu$ iff $\mu(U) \leq \nu(U)$ for all $U \in \Sigma_Y$.

Finite traces and executions

In [8], the authors consider coalgebras (X, γ) of endofunctors of the form $T \circ F$ with the monad $T : \mathsf{Set} \to \mathsf{Set}$ and the endofunctor $F : \mathsf{Set} \to \mathsf{Set}$ being related by a distributive law $\lambda : F \circ T \Rightarrow T \circ F$, and with the Kleisli category of T being *DCpo$_\perp$-enriched*; that is, in addition to *DCpo*-enrichedness, the orders on $\mathsf{Kl}(T)(X, Y)$ are required to have a bottom element. In this setting, the elements of the carrier I_F of the initial F-algebra provide the potential *finite traces* of states of $T \circ F$-coalgebras, and a *finite trace map* $ftr_\gamma : X \to T(I_F)$ is defined via finality in $\mathsf{Kl}(T)$. The crucial observation is that the initial F-algebra in Set lifts to a final $\overline{F}$-coalgebra in $\mathsf{Kl}(T)$ (where, as before, $\overline{F} : \mathsf{Kl}(T) \to \mathsf{Kl}(T)$ is the lifting of F to $\mathsf{Kl}(T)$ induced by λ). Thus, the finite trace map arises as the unique coalgebra morphism from the $\overline{F}$-coalgebra in $\mathsf{Kl}(T)$ induced by a $T \circ F$-coalgebra in Set to the final $\overline{F}$-coalgebra. The resulting notion of trace of a state of a $T \circ F$-coalgebra is referred to as *fat trace* in [11], as it retains the structure specified by the transition type F and therefore may involve branching.

A *finite execution map* for a $T \circ F$-coalgebra (X, γ) is also defined in [11], as the finite trace map obtained by regarding (X, γ) as a $T \circ F \circ (X \times \mathsf{Id})$-coalgebra. Here we propose a variant of this notion obtained by replacing the functor $F \circ (X \times \mathsf{Id})$ with the functor $X \times F$. The reason for this variation is that we expect finite executions starting in a state of a coalgebra to incorporate the state itself.

Definition 2.3 Let $T : \mathsf{C} \to \mathsf{C}$ be a strong monad, $F : \mathsf{C} \to \mathsf{C}$ be an endofunctor, and $\lambda : F \circ T \Rightarrow T \circ F$ be a distributive law of T over F. Also, for a $T \circ F$-coalgebra (X, γ), let (I_X, ι_X) denote an initial $(X \times F)$-algebra, and let $\lambda_X : (X \times F) \circ T \Rightarrow T \circ (X \times F)$ denote the natural transformation given by $(\lambda_X)_Y = st_{X,FY} \circ (id_X \times \lambda_Y)$. The *finite execution map*

$\mathsf{fexec}_\gamma : X \to TI_X$ is the C-map underlying the unique $\overline{X \times F}$-coalgebra morphism from $(X, st_{X,FX} \circ \langle id_X, \gamma \rangle)$ to the final $\overline{X \times F}$-coalgebra.

Modal logics for coalgebras

Our path-based coalgebraic temporal logics will be based on the notion of predicate lifting, as introduced by Pattinson [13]. However, the semantics of these logics will differ somewhat from the standard semantics of coalgebraic modal logics induced by predicate liftings, as defined e.g. in loc. cit. Also, the notion of predicate lifting used here is more general than the original one of [13], and applies to endofunctors on both Set and Meas.

We begin by fixing a category C with forgetful functor $U : \mathsf{C} \to \mathsf{Set}$, and a contravariant functor $P : \mathsf{C} \to \mathsf{Set}^{\mathrm{op}}$ such that P is a subfunctor of $\hat{\mathcal{P}} \circ U$, with $\hat{\mathcal{P}} : \mathsf{Set} \to \mathsf{Set}^{\mathrm{op}}$ the contravariant powerset functor. Thus, for each state space X, PX specifies a set of admissible predicates. As instances of P we will consider the contravariant powerset functor $\hat{\mathcal{P}} : \mathsf{Set} \to \mathsf{Set}^{\mathrm{op}}$ (in the case of coalgebras of endofunctors on Set), and the functor taking a measurable space to the carrier of its underlying σ-algebra (in the case of coalgebras of endofunctors on Meas).

Now given an endofunctor $F : \mathsf{C} \to \mathsf{C}$ and $n \in \omega$, an *n-ary predicate lifting for F* is a natural transformation $\lambda : P^n \Rightarrow P \circ F$. For ease of notation, we assume all predicate liftings to be unary, however, all our results generalise straightforwardly to predicate liftings with arbitrary finite arities. We briefly recall the syntax and semantics of coalgebraic modal logics induced by predicate liftings. Given a set Λ of predicate liftings for F, the modal language $\mathcal{L}_\Lambda$ has formulas given by:

$$\mathcal{L}_\Lambda \ni \varphi ::= \mathsf{tt} \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\lambda]\varphi \qquad (\lambda \in \Lambda)$$

A coalgebraic semantics for this language is obtained by defining $[\![\varphi]\!]_\gamma \subseteq PC$ for each F-coalgebra (C, γ), by structural induction on $\varphi \in \mathcal{L}_\Lambda$. The interesting case is $[\![[\lambda]\varphi]\!]_\gamma = (P\gamma)(\lambda_C([\![\varphi]\!]_\gamma))$ for $\varphi \in \mathcal{L}_\Lambda$ and $\lambda \in \Lambda$. In Section 4, we will see a novel use of modalities arising from predicate liftings, namely to interpret state formulas in path-based temporal logics. There, we will typically require our predicate liftings to be *monotone*, in that $A \subseteq B$ implies $\lambda_X(A) \subseteq \lambda_X(B)$ for all X and all $A, B \in PX$.

3 Possibly Infinite Traces and Executions

Our aim is to define a notion of possibly infinite execution of a state in a coalgebra, to be used in the semantics of path-based coalgebraic temporal logics. Some initial steps in this direction were made in [10], where a notion of infinite trace was defined for coalgebras of type $\mathcal{P} \circ F$, with $F : \mathsf{Set} \to \mathsf{Set}$

a polynomial functor equipped with a distributive law $\lambda : F \circ \mathcal{P} \Rightarrow \mathcal{P} \circ F$. Specifically, it was observed in loc. cit. that the final F-coalgebra in Set (whose elements represent potential infinite traces) gives rise to a weakly final $\overline{F}$-coalgebra in $\mathsf{Kl}(\mathcal{P})$. Then, for a $\mathcal{P} \circ F$-coalgebra, an infinite trace map was obtained using weak finality, by regarding this coalgebra as an $\overline{F}$-coalgebra in $\mathsf{Kl}(\mathcal{P})$. The order-enrichedness of $\mathsf{Kl}(\mathcal{P})$ guaranteed the existence of a canonical choice for the infinite trace map.

Here we propose a notion of infinite trace that applies to coalgebraic signatures of the form $T \circ F$, with T a monad and F an endofunctor on a category C, related through a distributive law of T over F and subject to some additional constraints. Throughout this section, C denotes a category with countable limits, $F : \mathsf{C} \to \mathsf{C}$ is an endofunctor, $T : \mathsf{C} \to \mathsf{C}$ is a strong monad whose Kleisli category is *DCpo*-enriched, and $\lambda : F \circ T \Rightarrow T \circ F$ is a distributive law of T over F.

3.1 Possibly infinite traces

As in [10], the final F-coalgebra provides the potential infinite traces of elements of $T \circ F$-coalgebras. We work under the assumption that F preserves the limit of the following ω^{op}-chain

$$1 \xleftarrow{!} F1 \xleftarrow{F!} F^2 1 \xleftarrow{F^2!} \cdots$$

with 1 a final object in C and $! : F1 \to 1$ the unique such map[4]. Assuming the above, the carrier of the final F-coalgebra is obtained as the limit in C of the above ω^{op}-chain. We let $(Z, \zeta : Z \to FZ)$ denote a final F-coalgebra, and write $\pi_i : Z \to F^i 1$ with $i \in \omega$ for the corresponding projections. We begin by showing that, under some additional constraints on the monad T, a $T \circ F$-coalgebra $\gamma : X \to TFX$ induces a cone over the ω^{op}-chain:

$$T1 \xleftarrow{T!} TF1 \xleftarrow{TF!} TF^2 1 \xleftarrow{TF^2!} \cdots$$

We define an ω-indexed family of maps $(\gamma_i : X \to TF^i 1)_{i \in \omega}$ by:

- $\gamma_0 = \eta_1 \circ !_X : X \to T1$, where $!_X : X \to 1$ is the unique such map,
- $\gamma_{i+1} = \mu_{F^{i+1}1} \circ T\lambda_{F^i 1} \circ TF\gamma_i \circ \gamma : X \to TF^{i+1}1$ for $i \in \omega$.

That is, the maps γ_i arise by unfolding the coalgebra structure i times, and using the distributive law λ of T over F and the monad multiplication to discard inner occurrences of T from the codomain of the maps γ_i. As the elements of $F^i 1$ define finite approximations of potential infinite traces, the maps γ_i can be regarded as providing finite approximations of the infinite

[4] This assumption is weaker than requiring F to preserve the limits of *all* ω^{op}-chains, a condition that will not hold for certain instances of F considered later in the paper.

trace map for the $T \circ F$-coalgebra γ. It is also worth noting that one can alternatively define the γ_is as maps in $\mathsf{Kl}(T)$:

- $\gamma_0 = J!_X$,
- $\gamma_{i+1} = \overline{F}\gamma_i \circ \gamma$ for $i \in \omega$.

Lemma 3.1 *Let $!_{TF1} : TF1 \to 1$ be the only such map. If $\eta_1 \circ !_{TF1} = T!$, then the above γ_is define a cone over the ω^{op}-chain $(JF^i!)_{i\in\omega}$ in $\mathsf{Kl}(T)$.*

Proof. The hypothesis ensures that $\gamma_0 = J! \circ \gamma_1$. Now assuming $\gamma_i = JF^i! \circ \gamma_{i+1}$, we immediately obtain $\overline{F}\gamma_i = \overline{F}JF^i! \circ \overline{F}\gamma_{i+1} = JF^{i+1}! \circ \overline{F}\gamma_{i+1}$, where the last equality follows by Lemma 2.2. Precomposition with γ finally gives $\gamma_{i+1} = JF^{i+1}! \circ \gamma_{i+2}$.

We immediately observe that the hypothesis of the above result is *not* satisfied by either of the two monads identified earlier:

- for $T = \mathcal{P}$, $(\eta_1 \circ !_{TF1})(\emptyset) = 1 \neq \emptyset = (\mathcal{P}!)(\emptyset)$;
- for $T = \mathcal{G}$, $(\eta_1 \circ !_{TF1})(\nu_0) = \mu_1 \neq \mu_0 = (\mathcal{G}!)(\nu_0)$, where ν_0 is the subprobability measure on $F(1, \mathcal{P}1)$[5] which assigns the value 0 to each measurable set, whereas μ_0 and μ_1 are the subprobability measures on $(1, \mathcal{P}1)$ given by $\mu_0(1) = 0$ and respectively $\mu_1(1) = 1$.

To remedy the situation, we will work with submonads of these two monads for which the hypothesis of Lemma 3.1 is true. To this end, we first note that if the monad T is such that $\eta_1 : 1 \to T1$ is an isomorphism, then the equality required by Lemma 3.1 is obtained immediately by finality. Strong monads with the above property are called *affine*, see e.g. [9] for an overview. Moreover, [9] shows how to construct, for any strong monad T, its *affine submonad* T_a, which is itself commutative whenever T is. This construction yields:

- the non-empty powerset monad $\mathcal{P}^+ : \mathsf{Set} \to \mathsf{Set}$ as the affine part of $\mathcal{P}$,
- the probability measure monad $\mathcal{G}_1 : \mathsf{Meas} \to \mathsf{Meas}$ (with $\mathcal{G}_1(X, \Sigma_X)$ containing only the probability measures on (X, Σ_X)) as the affine part of $\mathcal{G}$.

Thus, for $T = \mathcal{P}^+$ and $T = \mathcal{G}_1$, Lemma 3.1 applies. We also note that the canonical distributive laws of the original monads ($\mathcal{P}$, respectively $\mathcal{G}$) restrict to distributive laws of their affine submonads, and that the Kleisli categories of the affine submonads inherit an order-enriched structure from the Kleisli categories of the original monads. For the latter statement, one must verify that joins (taken in $\mathsf{Kl}(T)(X, Y)$) of directed sets in $\mathsf{Kl}(T_a)(X, Y)$ are themselves elements of $\mathsf{Kl}(T_a)(X, Y)$ and are preserved by arrow composition; this is straightforward in both cases. In fact, for $T = \mathcal{G}_1$, the inherited order on $\mathsf{Kl}(\mathcal{G}_1)(X, Y)$ is the equality. The former statement follows from a general result stating that any distributive law of a strong monad T over an endofunctor

[5] Note that $(1, \mathcal{P}1)$ is a final object in Meas.

F restricts to a distributive law of T_a over F.

Proposition 3.2 *Let $\lambda : F \circ T \Rightarrow T \circ F$ be a distributive law of T over F. Then, λ restricts to a distributive law $\lambda : F \circ T_a \Rightarrow T_a \circ F$.*

Proof. As shown in [9], the action of the monad T_a on a C-object X is given by the following pullback diagram:

$$\begin{array}{ccc} T_aX & \xrightarrow{\iota_X} & TX \\ {\scriptstyle !_{T_aX}}\downarrow & & \downarrow{\scriptstyle T!_X} \\ 1 & \xrightarrow{\eta_1} & T1 \end{array}$$

Thus, using that $!_{F1} \circ F!_X = !_{FX}$ (by finality of 1), the pullback diagram defining T_aFX can be written as

$$\begin{array}{ccc} T_aFX & \xrightarrow{\iota_{FX}} & TFX \\ \vdots & & \downarrow{\scriptstyle TF!_X} \\ \vdots & & TF1 \\ \vdots & & \downarrow{\scriptstyle T!_{F1}} \\ \downarrow & \xrightarrow{\eta_1} & T1 \end{array}$$

Next, note that the maps $\lambda_X \circ F\iota_X : FT_aX \to TFX$ and $!_{F1} \circ F!_{T_aX} : FT_aX \to 1$ define a cone over the diagram given by $T!_{F1} \circ TF!_X$ and η_1:

$$\begin{array}{rll} T!_{F1} \circ TF!_X \circ \lambda_X \circ F\iota_X = & & \text{(naturality of } \lambda) \\ T!_{F1} \circ \lambda_1 \circ FT!_X \circ F\iota_X = & & \text{(definition of } T_aX) \\ T!_{F1} \circ \lambda_1 \circ F\eta_1 \circ F!_{T_aX} = & & \text{(compatibility of } \lambda \text{ with monad structure)} \\ T!_{F1} \circ \eta_{F1} \circ F!_{T_aX} = & & \text{(naturality of } \eta) \\ \eta_1 \circ !_{F1} \circ F!_{T_aX} & & \end{array}$$

The definition of T_aFX now yields a map $(\lambda_a)_X : FT_aX \to T_aFX$. The naturality of the resulting maps and their compatibility with the monad structure follow easily by diagram chasing.

For our two examples ($T = \mathcal{P}^+$ and $T = \mathcal{G}_1$), assuming that F is a shapely polynomial functor, one can simply work with the canonical distributive laws. An easy induction proof (not given here) shows that these coincide with the distributive laws given by the previous result. However, Proposition 3.2 shows how to obtain a distributive law of the affine submonad over an *arbitrary* endofunctor.

To motivate our definition of the infinite trace map of a $T \circ F$-coalgebra (X, γ), let us examine the case $T = \mathcal{P}^+$. Since the map γ_i takes a state of the coalgebra to a set of i-depth approximations of its possibly infinite traces, it seems natural to define the infinite trace map as a function $tr_\gamma : X \to \mathcal{P}^+Z$

sending a state s of the coalgebra to the set of possibly infinite traces whose i-depth approximation belongs to $\gamma_i(s)$. Such a trace map can be defined by exploiting the weak preservation of limits of ω^{op}-chains by J (which, in turn, follows from the weak preservation of such limits by $\mathcal{P}^+$). However, this property only guarantees the *existence* of a mediating map $tr_\gamma : X \to JZ$ in $\mathsf{Kl}(T)$. As shown in [10] for the case $T = \mathcal{P}$, a canonical choice for the infinite trace map is provided by the *largest* mediating map. Its existence is here guaranteed by the *DCpo*-structure of $\mathsf{Kl}(\mathcal{P}^+)(X, Z)$, together with the observation that in this particular case the mediating maps form a directed set. This justifies the following general definition of the infinite trace map.

Definition 3.3 Assume that the monad T is affine and that the functor J weakly preserves the limit $(Z, (\pi_i)_{i\in\omega})$ of the ω^{op}-chain $(F^i!)_{i\in\omega}$. For a $T \circ F$-coalgebra (X, γ), let $(X, (\gamma_i : X \to JF^i1)_{i\in\omega})$ be the induced cone over $(JF^i!)_{i\in\omega}$, and assume further that the corresponding mediating maps form a directed set. The *possibly infinite trace map* is the largest [6] mediating map $tr_\gamma : X \to JZ$ arising from the weak limiting property of $(JZ, (J\pi_i)_{i\in\omega})$ (regarded as a map in C).

In particular, Definition 3.3 can be applied to the non-empty powerset monad $\mathcal{P}^+ : \mathsf{Set} \to \mathsf{Set}$, as well as to the probability measure monad $\mathcal{G}_1 : \mathsf{Meas} \to \mathsf{Meas}$. The resulting notions of infinite trace are discussed in Sections 3.3 and 3.4. We also note that the affine submonad of the lift monad $1 + \mathsf{Id}$ on Set (as considered in [8]) is the identity monad, to which Definition 3.3 applies trivially. A treatment of monads that are *not* affine is outside the scope of this paper.

We conclude this section by proving some properties of the infinite trace map, similar to the *defining* properties of the infinite trace map in [10].

Proposition 3.4 *Under the assumptions of Definition 3.3, the trace map $tr_\gamma : X \to JZ$ defines an* op-lax *$\overline{F}$-coalgebra morphism from (X, γ) to $(JZ, J\zeta)$, that is, $\overline{F}tr_\gamma \circ \gamma \sqsubseteq J\zeta \circ tr_\gamma$. Under the additional assumptions that $(JZ, (J\pi_i)_{i\in\omega})$ is a limit of $(JF^i!)_{i\in\omega}$, tr_γ defines an $\overline{F}$-coalgebra morphism, that is, $\overline{F}tr_\gamma \circ \gamma = J\zeta \circ tr_\gamma$.*

Proof. We begin by noting that the final F-coalgebra $\zeta : Z \to FZ$ satisfies $F\pi_i \circ \zeta = \pi_{i+1}$ for all $i \in \omega$, and hence, in $\mathsf{Kl}(T)$ we have $JF\pi_i \circ J\zeta = J\pi_{i+1}$ for all $i \in \omega$. Now recall that $(JFZ, (JF\pi_i)_{i\in\omega})$ is a weak limit of $(JF^{i+1}!)_{i\in\omega}$. Moreover, since $J\zeta$ is an isomorphism in $\mathsf{Kl}(T)$ (and hence admits an inverse), and since arrow composition in $\mathsf{Kl}(T)$ preserves directed joins, it follows that the map $J\zeta \circ tr_\gamma : X \to JFZ$ is the largest mediating map for the cone $(X, (\gamma_{i+1})_{i\in\omega})$ over the ω^{op}-chain $(JF^{i+1}!)_{i\in\omega}$. On the other hand, we have: $JF\pi_i \circ \overline{F}tr_\gamma \circ \gamma = \overline{F}J\pi_i \circ \overline{F}tr_\gamma \circ \gamma = \overline{F}\gamma_i \circ \gamma = \gamma_{i+1}$. Hence, since

[6] w.r.t. the order on $\mathsf{Kl}(T)(X, Z)$

$J\zeta \circ tr_\gamma : X \to JFZ$ is the largest mediating map for $(X, (\gamma_{i+1})_{i\in\omega})$, we obtain $\overline{F}tr_\gamma \circ \gamma \sqsubseteq J\zeta \circ tr_\gamma$. That is, tr_γ defines an op-lax $\overline{F}$-coalgebra morphism from (X, γ) to $(JZ, J\zeta)$. Under the stronger assumption that $(JZ, (J\pi_i)_{i\in\omega})$ is a limit of $(JF^i!)_{i\in\omega}$, uniqueness of a mediating arrow induced by the cone $(X, (\gamma_{i+1})_{i\in\omega})$ over $(JF\pi_i)_{i\in\omega}$ yields $\overline{F}tr_\gamma \circ \gamma = J\zeta \circ tr_\gamma$, that is, tr_γ defines an $\overline{F}$-coalgebra morphism.

In the case of the non-empty powerset monad, the above result only implies that the infinite trace map is an *op-lax* coalgebra morphism. This is weaker than the defining property of the infinite trace map in [10], which asks for a proper coalgebra morphism. The study of sufficient conditions for the infinite trace map to define a proper coalgebra morphism for an arbitrary (affine) monad T remains an open question, but we conjecture that the local continuity of the functor $\overline{F}$ [7] will be at least a necessary condition.

On the other hand, we will see later that the additional assumption of Proposition 3.4 which ensures that the trace map is a coalgebra morphism holds for the probability measure functor $\mathcal{G}_1$ on Meas, when taking certain shapely polynomial functors on Meas as instances of F.

3.2 Possibly infinite executions

To obtain a notion of possibly infinite execution of a state in a $T \circ F$-coalgebra, we use the approach in the previous section with a different choice of functor F. Similarly to Definition 2.3, given a $T \circ F$-coalgebra (X, γ), we consider the endofunctor $F_X : \mathsf{C} \to \mathsf{C}$ given by $F_X(Y) = X \times FY$ and the distributive law $\lambda_X : F_X \circ T \Rightarrow T \circ F_X$ given by $(\lambda_X)_Y = st_{X,FY} \circ (id_X \times \lambda_Y)$. We call an element of the carrier of the final F_X-coalgebra (Z_X, ζ_X) a potential *infinite execution*, or *computation path*.

Definition 3.5 Assume that T is affine and that J weakly preserves the limit $(Z_X, (\pi_i)_{i\in\omega})$ of the ω^{op}-chain $({F_X}^i!)_{i\in\omega}$. For a $T \circ F$-coalgebra (X, γ), let $(X, (\gamma_i : X \to J{F_X}^i 1)_{i\in\omega})$ be the cone over $(J{F_X}^i!)_{i\in\omega}$ induced by the $T \circ F_X$-coalgebra $(X, st_{X,FX} \circ \langle id_X, \gamma\rangle)$, and assume that the corresponding mediating maps form a directed set. The *possibly infinite execution map* $\mathsf{exec}_\gamma : X \to JZ_X$ of (X, γ) is the possibly infinite trace map of the $T \circ F_X$-coalgebra $(X, st_{X,FX} \circ \langle id_X, \gamma\rangle)$.

3.3 (Labelled) transition systems

These are modelled as $\mathcal{P}^+ \circ F$-coalgebras, with $F = \mathsf{Id}$ (respectively $F = A \times \mathsf{Id}$ for a fixed set A of labels). Our use of the non-empty powerset monad agrees

[7] A functor $F : \mathsf{C} \to \mathsf{D}$ between *DCpo*-enriched categories is *locally continuous* if it preserves suprema of directed joins in $\mathsf{C}(X, Y)$ for each X, Y. In enriched categorical terms, F is a *DCpo*-enriched functor.

with the standard constraint put on transition systems when defining computation paths. The next result ensures that the hypotheses of Definitions 3.3 and 3.5 are satisfied.

Lemma 3.6 *The (non-empty) powerset functor weakly preserves limits of ω^{op}-chains; hence, by Lemma 2.1, so does J. Moreover, the resulting mediating maps, regarded as arrows in $\mathsf{Kl}(\mathcal{P})$ (resp. $\mathsf{Kl}(\mathcal{P}^+)$) form a directed set.*

Proof. Let $(Z, (\pi_i)_{i\in\omega})$ denote the limit of an ω^{op}-chain $(f_i : Z_{i+1} \to Z_i)_{i\in\omega}$. For a cone $(\gamma_i : X \to \mathcal{P}Z_i)_{i\in\omega}$ over $(\mathcal{P}f_i : \mathcal{P}Z_{i+1} \to \mathcal{P}Z_i)_{i\in\omega}$, the map $m : X \to \mathcal{P}Z$ given by $m(x) = \{z \in Z \mid \pi_i(z) \in \gamma_i(x) \text{ for } i \in \omega\}$ for $x \in X$ is a mediating map. (If $X = \emptyset$, the existence of a mediating map is trivial.) The same applies when replacing $\mathcal{P}$ by $\mathcal{P}^+$. This time, one also has to show that the set defining $m(x)$ is non-empty. Using the axiom of choice one can construct, for each $x \in X$, a sequence $(z_i)_{i\in\omega}$ with $z_i \in \gamma_i(x)$ and $f_i(z_{i+1}) = z_i$ for $i \in \omega$; this, in turn, yields $z \in Z$ with $\pi_i(z) \in \gamma_i(x)$ for $i \in \omega$.

For the second statement, note that the mediating map m defined previously is above any other mediating map (under the inclusion order), and thus the set of mediating maps is directed.

Remark 3.7 To see that neither $\mathcal{P}$ nor $\mathcal{P}^+$ preserve limits of ω^{op}-chains, consider the final sequence $(f_i : Z_{i+1} \to Z_i)_{i\in\omega}$ of the endofunctor $1 + A \times \mathsf{Id}$, with $Z_i = \bigcup_{0\le j\le i} A^i$, and with limit object $Z = A^* \cup A^\omega$. Now define a cone $(\gamma_i : 1 \to \mathcal{P}Z_i)_{i\in\omega}$ by letting $\gamma_i(*)$ consist only of the i-long sequence of a's, for some fixed $a \in A$. Then, both $m(*) = \{a\}^*$ and $m'(*) = \{a\}^* \cup \{a\}^\omega$ define mediating maps. (A similar example is discussed in [8, Section 4.2].)

As a result of Lemma 3.6, Definition 3.5 yields, for each state in a transition system, a set of infinite executions. As expected, this coincides with the set of computation paths from that state, as considered in the semantics of CTL*. For $F = A \times \mathsf{Id}$, the infinite execution map gives, for each state s, the set of labelled computation paths from s, as infinite sequences of the form $s = s_0 a_1 s_1 a_2 s_2 \ldots$ with $s_i \xrightarrow{a_i} s_{i+1}$ for $i \in \omega$, whereas the infinite trace map yields the sequences of labels that occur along such labelled computation paths.

One can also vary the functor F in order to model explicit termination. This is achieved by taking $F = 1 + \mathsf{Id}$ or $F = 1 + A \times \mathsf{Id}$, as in [8]. In these cases, the resulting possibly infinite trace (execution) maps capture both finite and infinite traces (respectively computation paths).

3.4 Probabilistic models

A large variety of discrete probabilistic models have been studied, see e.g. [17] for a coalgebraic account of such models. Among these, probabilistic transition systems (also called *Markov chains*) appear as coalgebras of the endofunctor

$\mathcal{D} = \mathcal{D} \circ \mathsf{Id}$ and are used to interpret the logic PCTL [7], while *generative probabilistic systems* coincide with $\mathcal{D} \circ (A \times \mathsf{Id})$-coalgebras. Here, $\mathcal{D} : \mathsf{Set} \to \mathsf{Set}$ denotes the *probability distribution monad*, a submonad of the subprobability distribution monad defined on objects by $\mathcal{D}X = \{\mu \in \mathcal{S}X \mid \sum_{x \in X} \mu(x) = 1\}$.

Unfortunately, although affine, the monad $\mathcal{D}$ does not satisfy the requirement of Definition 3.3 concerning the weak preservation of limits by the induced functor J. To see this, let $F : \mathsf{Set} \to \mathsf{Set}$ be given by $FX = \{a, b\} \times X$, and $\mu_i \in \mathcal{D}F^i 1$ be given by $\mu_i(x) = \frac{1}{2^i}$ for $x \in \{a, b\}^i$, with $i \in \omega$. Thus, each μ_i defines a *finite* probability distribution over $F^i 1$, and we have $(\mathcal{D}^i !)(\mu_{i+1}) = \mu_i$ for $i \in \omega$. However, there is no probability distribution μ on the final F-coalgebra (whose carrier, $\{a, b\}^\omega$, is uncountable) such that $(\mathcal{D}\pi_i)(\mu) = \mu_i$ for $i \in \omega$ – any such μ could only take non-zero values on countably-many elements of Z. Indeed, a state of a $\mathcal{D} \circ F$-coalgebra will in general have uncountably many infinite traces, and the emphasis when defining an infinite trace map should be on measuring *sets* of traces rather than individual traces.

A satisfactory treatment of infinite traces for discrete probabilistic models turns out to be possible by regarding such models as coalgebras of the probability measure monad $\mathcal{G}_1$. For technical reasons that will soon become clear, we will work in a subcategory of Meas, namely the full subcategory SB of Meas whose objects are *standard Borel spaces* (spaces whose measurable sets arise as the Borel sets induced by a complete, separable metric, see e.g. [4]). A notable property of this category is that it is closed under countable coproducts and countable limits in Meas (see e.g. [16, Fact 1]). We also note that a discrete measurable space $(X, \mathcal{P}X)$ is standard Borel if and only if X is countable. As a result, we will only be able to define notions of infinite trace and infinite execution for $\mathcal{D} \circ F$-coalgebras *with countable carrier*. We will do so by lifting the functor F to a functor $\hat{F} : \mathsf{SB} \to \mathsf{SB}$, and regarding a $\mathcal{D} \circ F$-coalgebra on Set as a $\mathcal{G}_1 \circ \hat{F}$-coalgebra on SB.

We now proceed to define a restricted version of shapely polynomial functors on Meas. The restriction is driven by the need to work in the subcategory SB of Meas. Specifically, we call an endofunctor on Meas a *restricted shapely polynomial functor* if it is built from identity and constant functors $C_{(X, \Sigma_X)}$ with (X, Σ_X) a standard Borel space, using finite products and *countable* coproducts. Then, given a *restricted* shapely polynomial functor F on Set, that is, a functor built from identity and *countable* constant functors using finite products and *countable* coproducts, we write $\hat{F} : \mathsf{Meas} \to \mathsf{Meas}$ for its counterpart on measurable spaces, defined by structural induction on F:

- $\widehat{\mathsf{Id}}$ is the identity functor on Meas,
- $\widehat{C_X}$ is the constant functor $C_{(X, \mathcal{P}X)}$, for each countable set X,
- $\widehat{F_1 \times F_2} = \widehat{F_1} \times \widehat{F_1}$,

- $\widehat{\coprod_{i\in\omega} F_i} = \coprod_{i\in\omega} \widehat{F}_i$.

Lemma 3.8 *If $F : \mathsf{Set} \to \mathsf{Set}$ is a restricted shapely polynomial functor, then so is $\widehat{F} : \mathsf{Meas} \to \mathsf{Meas}$. Moreover, $\widehat{F}$ preserves (discrete) SB-spaces.*

Proof. The first statement is immediate. Preservation of SB-spaces by $\widehat{F}$ follows from results in [16], whereas preservation of discrete spaces follows by induction on the structure of F:

- For $F = C_X$ with X countable, $\widehat{F}(Y, \mathcal{P}Y) = (X, \mathcal{P}X) = (FX, \mathcal{P}FX)$ for all Y.
- For $F = \mathsf{Id}$, $\widehat{Id}(X, \mathcal{P}X) = (X, \mathcal{P}X) = (FX, \mathcal{P}FX)$ for all X.
- For $F = F_1 \times F_2$, $\widehat{F}(X, \mathcal{P}X) = \widehat{F_1}(X, \mathcal{P}X) \times \widehat{F_2}(X, \mathcal{P}X) = (F_1X, \mathcal{P}F_1X) \times (F_2X, \mathcal{P}F_2X) = (F_1X \times F_2X, \mathcal{P}(F_1X \times F_2X))$, where the last equality follows from finite products of discrete SB-spaces being themselves discrete SB-spaces.
- The case $F = \coprod_{i\in\omega} F_i$ is treated similarly.

As a result, we immediately obtain

Proposition 3.9 *A $\mathcal{D} \circ F$-coalgebra (X, γ) with countable carrier yields a $\mathcal{G}_1 \circ \widehat{F}$-coalgebra $((X, \mathcal{P}X), \widehat{\gamma})$, such that the cone $(\gamma_i)_{i\in\omega}$ in $\mathsf{Kl}(\mathcal{D})$:*

$$\begin{array}{ccccccccc} X & & & & & & & & \\ {\scriptstyle\gamma_0}\downarrow & \searrow^{\gamma_1} & & \searrow^{\gamma_2} & & & & & \\ J1 & \xleftarrow[J!]{} & JF1 & & \xleftarrow[JF!]{} & JF^2 1 & & \xleftarrow[JF^2!]{} & \cdots \end{array}$$

with the γ_is being as in Section 3.1, defines a cone in $\mathsf{Kl}(\mathcal{G}_1)$:

$$\begin{array}{ccccccccc} (X, \mathcal{P}X) & & & & & & & & \\ {\scriptstyle\gamma_0}\downarrow & \searrow^{\gamma_1} & & \searrow^{\gamma_2} & & & & & \\ J'(1, \mathcal{P}1) & \xleftarrow[J'!]{} & J'\widehat{F}(1, \mathcal{P}1) & & \xleftarrow[J'\widehat{F}!]{} & J'\widehat{F}^2(1, \mathcal{P}1) & & \xleftarrow[J'\widehat{F}^2!]{} & \cdots \end{array}$$

where $J : \mathsf{Set} \to \mathsf{Kl}(\mathcal{D})$ and $J' : \mathsf{Meas} \to \mathsf{Kl}(\mathcal{G}_1)$ are as in Section 2.

The coalgebra map $\widehat{\gamma} : (X, \mathcal{P}X) \to \mathcal{G}_1\widehat{F}(X, \mathcal{P}X) = \mathcal{G}_1(FX, \mathcal{P}FX)$ yields, for each state $x \in X$, the probability measure on $(FX, \mathcal{P}FX)$ induced by the probability distribution $\gamma(x)$ on FX. Since $(1, \mathcal{P}1)$ is final in Meas, the latter of the above cones is over the image under J' of the final sequence of $\widehat{F}$. As a result, we can use the existence of trace maps of $\mathcal{G}_1 \circ \widehat{F}$-coalgebras to define trace maps for $\mathcal{D} \circ F$-coalgebras.

The next lemma ensures that $\widehat{F}$ and $\widehat{F_X}$ preserve the limit of the initial ω^{op}-segment of their respective final sequences, as required by Definitions 3.3 and 3.5.

Lemma 3.10 ([16]) *Restricted shapely polynomial functors on* Meas *preserve surjective* SB*-morphisms and limits of* ω^{op}*-chains of surjective* SB*-morphisms.*

Proof. It was proved in [16, Proposition 3] that the class of endofunctors on Meas that preserve surjective SB-morphisms and limits of ω^{op}-chains of surjective SB-morphisms is closed under countable coproducts and countable limits. The conclusion then follows after noting that the identity functor and constant functors $C_{(X,\Sigma_X)}$ with (X, Σ_X) a standard Borel space belong to this class.

The required property of $\widehat{F}$ now follows, since $! : \widehat{F}(1, \mathcal{P}1) \to (1, \mathcal{P}1)$ is a surjective SB-morphism (assuming that F is non-trivial, i.e. $F1 \neq \emptyset$). As a result, for every restricted shapely polynomial functor F on Set, the final sequence of $\widehat{F}$ belongs to SB, stabilises at ω, and its limit is the carrier of a final $\widehat{F}$-coalgebra, itself in SB. Moreover, if X is a countable set, the above also applies to the functor $F_X : \mathsf{Set} \to \mathsf{Set}$ defined by $F_X Y = X \times FY$. The restriction to countable carriers is necessary to ensure applicability of Definition 3.5. This is the reason for working with the category SB.

Recall from Section 2 that commutative monads on any category with products and coproducts admit canonical distributive laws over shapely polynomial functors. This applies in particular to the monad $\mathcal{G}_1$ and any restricted shapely polynomial functor on Meas. Then, to be able to apply Definition 3.1 to the functors $\widehat{F}$ and $\widehat{F_X}$, with $F : \mathsf{Set} \to \mathsf{Set}$ a restricted shapely polynomial functor and X a countable set, all that remains to verify is that the functor $\mathcal{G}_1$ weakly preserves the limits of the final sequences of $\widehat{F}$ and $\widehat{F_X}$. In fact, a stronger result holds:

Lemma 3.11 ([16]) *The functor* $\mathcal{G}_1 : \mathsf{Meas} \to \mathsf{Meas}$ *preserves limits of* ω^{op}*-chains of surjective* SB*-morphisms.*

We note that the result in [16] refers to the subprobability measure functor $\mathcal{G}$, but a similar proof can be given for the probability measure functor.

As a consequence, we obtain probabilistic trace and execution maps for $\mathcal{D} \circ F$-coalgebras with countable carrier, with $F : \mathsf{Set} \to \mathsf{Set}$ as above.

Definition 3.12 Let $F : \mathsf{Set} \to \mathsf{Set}$ be a restricted shapely polynomial functor, let (X, γ) be a $\mathcal{D} \circ F$-coalgebra with countable carrier, and let $(\gamma_i : (X, \mathcal{P}X) \to J'\widehat{F}^i(1, \mathcal{P}1))_{i \in \omega}$ denote the cone over $(J'\widehat{F}^i!)_{i \in \omega}$ induced by the $\mathcal{G}_1 \circ \widehat{F}$-coalgebra $\widehat{\gamma} : (X, \mathcal{P}X) \to \mathcal{G}_1\widehat{F}(X, \mathcal{P}X)$. The *probabilistic trace map* $tr_\gamma : X \to JZ$ is defined as the underlying function of the unique measurable map arising from the limiting property of $J'(Z, \Sigma_Z)$, where (Z, Σ_Z) is the carrier of a final $\widehat{F}$-coalgebra.

Since limits in Meas are constructed from the underlying limits in Set (see e.g. [15]), the state space Z of the final $\widehat{F}$-coalgebra is the carrier of a final

F-coalgebra, and thus the probabilistic trace map yields, as expected, for each state of a $\mathcal{D} \circ F$-coalgebra, a probability measure over (Z, Σ_Z).

Returning to the example of Markov chains ($F = \mathsf{Id}$), the resulting notion of probabilistic execution gives, for each state in a Markov chain, a probability measure over its computation paths. Similarly, in the case of generative probabilistic systems ($F = A \times \mathsf{Id}$), the notion of probabilistic execution gives, for each state, a probability measure over its labelled computation paths. Finally, explicit termination can be modelled by taking $F = 1 + \mathsf{Id}$ or $F = 1 + A \times \mathsf{Id}$, as in [8], and the resulting notions of possibly infinite execution also incorporate finite (labelled) computation paths.

4 Path-Based Coalgebraic Temporal Logics

We now introduce CTL*-like coalgebraic temporal logics whose semantics is defined in terms of possibly infinite executions. Throughout this section, we fix a monad $T : \mathsf{C} \to \mathsf{C}$, a functor $F : \mathsf{C} \to \mathsf{C}$, and a $T \circ F$-coalgebra (X, γ) together with a map $\mathsf{exec}_\gamma : X \to TZ_X$ obtained using the approach in Section 3, where (Z_X, ζ_X) is a final F_X-coalgebra. We note in passing that the temporal languages defined in this section can also be interpreted by using the finite execution map $\mathsf{fexec}_\gamma : X \to TI_X$ with (I_X, ι_X) an initial $(X \times F)$-algebra, as given by Definition 2.3, instead of the infinite execution map – the forthcoming definitions do not rely on the finality of (Z_X, ζ_X). However, this is only useful when $F0 \neq 0$, with 0 an initial object in C, as otherwise the initial F_X-algebra has empty carrier. In particular, modelling explicit termination via functors such as $F = 1 + \mathsf{Id}$ or $F = 1 + A \times \mathsf{Id}$ yields non-trivial finite execution maps to which the definitions in this section can be applied.

The temporal logics that we define are parameterised by sets Λ_F and Λ of monotone predicate liftings for the functors F and respectively T. The category C will be instantiated to Set and Meas.

We recall that the definition of predicate liftings requires functors $U : \mathsf{C} \to \mathsf{Set}$ and $P : \mathsf{C} \to \mathsf{Set}^{\mathrm{op}}$ such that P is a subfunctor of $\hat{\mathcal{P}} \circ U$. In addition, defining the semantics of path-based temporal logics will at least require that PX is closed under countable (including finite) unions and intersections, for each C-object X.

4.1 Path-based fixpoint logics

We first consider the case when PX is a complete lattice for each X. Under this assumption, which holds e.g. when $\mathsf{C} = \mathsf{Set}$ and $P = \hat{\mathcal{P}}$, we are able to define path-based coalgebraic *fixpoint* logics. These logics are two-sorted, with *path formulas* denoted by $\varphi^F, \psi^F, \ldots$ expressing properties of possibly infinite executions, and *state formulas* denoted by $\varphi, \psi, \ldots$ expressing properties of

states of $T \circ F$-coalgebras.

The language $\mu\mathcal{L} ::= \mu\mathcal{L}_{\Lambda}^{\Lambda_F}(\mathcal{V}_F, \mathcal{V})$ over a 2-sorted set $(\mathcal{V}_F, \mathcal{V})$ of propositional variables (with sorts for paths and respectively states) is defined by the grammar

$$\begin{aligned} \mu\mathcal{L}_F \ni \varphi^F &::= \mathsf{tt} \mid \mathsf{ff} \mid p^F \mid \varphi \mid \varphi^F \wedge \varphi^F \mid \varphi^F \vee \varphi^F \mid [\lambda_F]\varphi^F \mid \eta p^F.\varphi^F \\ \mu\mathcal{L} \ni \varphi &::= \mathsf{tt} \mid \mathsf{ff} \mid p \mid [\lambda]\varphi^F \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \end{aligned}$$

where $p^F \in \mathcal{V}_F$, $p \in \mathcal{V}$, $\eta \in \{\mu, \nu\}$, $\lambda_F \in \Lambda_F$ and $\lambda \in \Lambda$. Thus, path formulas are constructed from propositional variables and state formulas using positive boolean operators, modal operators $[\lambda_F]$ and fixpoint operators, whereas state formulas are constructed from atomic propositions and modal formulas $[\lambda]\varphi_F$ with φ_F a *path* formula, using positive boolean operators. The modal operators $[\lambda_F]$ and $[\lambda]$ with $\lambda_F \in \Lambda_F$ and $\lambda \in \Lambda$ are thus both applied to path formulas, to obtain new path formulas and respectively *state* formulas. They are, however, of very different natures: while the operators $[\lambda_F]$ quantify over the *one-step* behaviour of computation paths, the operators $[\lambda]$ quantify over the (suitably structured) *long-term* computation paths from particular states. This is made precise in the formal semantics of $\mu\mathcal{L}_{\Lambda}^{\Lambda_F}(\mathcal{V}_F, \mathcal{V})$, as defined below.

Given a $T \circ F$-coalgebra (X, γ) and a 2-sorted valuation $V : (\mathcal{V}_F, \mathcal{V}) \to (PZ_X, PX)$ (interpreting path and state variables as sets of computation paths and respectively of states), the semantics $(\!|\varphi^F|\!)_{\gamma,V} \in PZ_X$ of path formulas $\varphi^F \in \mu\mathcal{L}_F$ and $[\![\varphi]\!]_{\gamma,V} \in PX$ of state formulas $\varphi \in \mu\mathcal{L}$ is defined inductively on the structure of φ^F and φ by:

$$\begin{aligned} (\!|p^F|\!)_{\gamma,V} &= V(p^F) \\ (\!|\varphi|\!)_{\gamma,V} &= P(\pi_1 \circ \zeta_X)([\![\varphi]\!]_{\gamma,V}) \\ (\!|[\lambda_F]\varphi^F|\!)_{\gamma,V} &= (P(\pi_2 \circ \zeta_X) \circ (\lambda_F)_{Z_X})((\!|\varphi^F|\!)_{\gamma,V}) \\ (\!|\mu p^F.\varphi^F|\!)_{\gamma,V} &= lfp((\varphi^F)_{p^F}^{\gamma,V}) \\ (\!|\nu p^F.\varphi^F|\!)_{\gamma,V} &= gfp((\varphi^F)_{p^F}^{\gamma,V}) \\ [\![p]\!]_{\gamma,V} &= V(p) \\ [\![[\lambda]\varphi^F]\!]_{\gamma,V} &= (P\mathsf{exec}_\gamma \circ \lambda_{Z_X})((\!|\varphi^F|\!)_{\gamma,V}) \end{aligned}$$

and the usual clauses for the boolean operators, where, for $p^F \in \mathcal{V}_F$, $(\varphi^F)_{p^F}^{\gamma,V} : PX \to PX$ denotes the monotone map defined by $(\varphi^F)_{p^F}^{\gamma,V}(Y) = (\!|\varphi^F|\!)_{\gamma,V'}$ with $V'(p^F) = Y$ and $V'(q) = V(q)$ for $q \neq p^F$, whereas $lfp(\_)$ and $gfp(\_)$ construct least and respectively greatest fixpoints. We note that the monotonicity of the predicate liftings in Λ^F and Λ together with the absence of negation in either path or state formulas ensure that the maps $(\varphi^F)_{p^F}^{\gamma,V} : PX \to PX$ are monotone, and hence, by the Knaster-Tarski theorem, admit least and greatest fixpoints. Let us now examine the definition of the semantics of $\mu\mathcal{L}_{\Lambda}^{\Lambda_F}(\mathcal{V}_F, \mathcal{V})$

in more detail:

- To define $(\!|\varphi|\!)_{\gamma,V} \in PZ_X$ from $[\![\varphi]\!]_{\gamma,V} \in PX$, one uses the inverse image of the map $\pi_1 \circ \zeta_X$ which extracts the first state of a computation path in Z_X:

$$Z_X \xrightarrow{\zeta_X} X \times FZ_X \xrightarrow{\pi_1} X$$

  This formalises the idea that a state formula φ (regarded as a path formula) holds in a path precisely when it holds in the first state of that path.
- To define $(\!|[\lambda_F]\varphi^F|\!)_{\gamma,V} \in PZ_X$ from $(\!|\varphi^F|\!)_{\gamma,V} \in PZ_X$, one first applies the relevant component of the predicate lifting λ_F to obtain a set of one-step F-observations of computation paths, and then uses the inverse image of the map $\pi_2 \circ \zeta_X$

$$Z_X \xrightarrow{\zeta_X} X \times FZ_X \xrightarrow{\pi_2} FZ_X$$

  (which extracts the one-step F-observation of a computation path in Z_X) to obtain a set of computation paths again. This is the standard interpretation of the modal operator $[\lambda_F]$ in the F-coalgebra $\pi_2 \circ \zeta_X$.
- Finally, to define $[\![[\lambda]\varphi^F]\!]_{\gamma,V} \in PX$ from $(\!|\varphi^F|\!)_{\gamma,V} \in PZ_X$, one first applies the relevant component of the predicate lifting λ to obtain a set of suitably-structured computation paths, and then uses the inverse image of the execution map to obtain a set of states:

$$PZ_X \xrightarrow{(\lambda)_{Z_X}} PTZ_X \xrightarrow{P\mathsf{exec}_\gamma} PX$$

Example 4.1 We are now able to recover the negation-free fragment of the logic CTL*[8] as a fragment of the path-based fixpoint logic obtained by taking $T = \mathcal{P}^+$, $F = \mathsf{Id}$, $\Lambda = \{\Box, \Diamond\}$ and $\Lambda_F = \{\bigcirc\}$, where the predicate liftings $\lambda_\Box, \lambda_\Diamond : \hat{\mathcal{P}} \Rightarrow \hat{\mathcal{P}} \circ \mathcal{P}^+$ and $\lambda_\bigcirc : \hat{\mathcal{P}} \Rightarrow \hat{\mathcal{P}} \circ \mathsf{Id}$ associated to these modalities are given by:

$$\begin{aligned}(\lambda_\Box)_X(Y) &= \{Z \in \mathcal{P}^+X \mid Z \subseteq Y\},\\ (\lambda_\Diamond)_X(Y) &= \{Z \in \mathcal{P}^+X \mid Z \cap Y \neq \emptyset\},\\ (\lambda_\bigcirc)_X(Y) &= Y.\end{aligned}$$

The choice of $\lambda_\Box$ and $\lambda_\Diamond$ as predicate liftings for $\mathcal{P}^+$ captures precisely the path quantifiers **A** and **E** of CTL*, whereas the $\bigcirc$ modality captures the **X** operator on paths. The remaining path operators of CTL* (**F**, **G** and **U**) can be encoded as fixpoint formulas. For example, the CTL* path formula $\varphi\,\mathbf{U}\,\psi$ can be encoded as $\mu X.(\psi \vee (\varphi \wedge \bigcirc X))$.

[8] The entire language can also be obtained, using an approach similar to that of Section 4.2.

Moreover, by varying the functor F to $A \times \mathsf{Id}$, we obtain an interesting variant of CTL* interpreted over labelled transition systems. For this, we take $\Lambda_F = \{a \mid a \in A\} \cup \{\bigcirc\}$, where the predicate liftings $\lambda_a : 1 \Rightarrow \hat{\mathcal{P}} \circ (A \times \mathsf{Id})$ with $a \in A$ and $\lambda_\bigcirc : \hat{\mathcal{P}} \Rightarrow \hat{\mathcal{P}} \circ (A \times \mathsf{Id})$ are given by:

$$(\lambda_a)_X(*) = \{a\} \times X,$$
$$(\lambda_\bigcirc)_X(Y) = A \times Y.$$

The resulting temporal language can easily express the property "a occurs along every computation path", namely as $\Box\mu X.(a \vee \bigcirc X)$. The reader should compare this to the formulation of the same property in the language obtained by adding fixpoints to the negation-free variant of Hennessy-Milner logic, namely as $\mu X.(\langle\_\rangle\mathsf{tt} \wedge [-a]X)$. Here, the formulas $\langle\_\rangle\varphi$ and $[-a]\varphi$ should be read as "there exists a successor state (reachable by *some* label) satisfying φ" and respectively "all states reachable by labels other than a satisfy φ". It is easy to see that, as the required nesting depth of fixpoint operators increases, the encodings of path properties in the latter language become complex very quickly, making the path-based language a better alternative.

4.2 Path-based temporal logics with Until operators

We now return to the more general situation when PX is only closed under countable unions and intersections. This is for instance the case when $\mathsf{C} = \mathsf{Meas}$ and $P(X, \Sigma_X) = \Sigma_X$. In this case, least or greatest fixpoints of monotone maps on PX do not necessarily exist, and we must restrict ourselves to temporal operators for which we are able to provide a well-defined semantics. In what follows we only consider Until operators similar to the ones of CTL* and PCTL, however, our approach supports more general temporal operators. In particular, a suitable choice of temporal operators can be used to obtain the full language of CTL* without resorting to arbitrary fixpoints.

Before defining the general syntax of path-based temporal logics with Until operators, we observe that the structure of the functor F may result in the associated notions of trace and execution involving some branching (as is for instance the case when $FX = A \times X \times X$). In such cases, Until operators must take into account the branching. Due to space limitations, here we only consider existential versions of branching Until operators, and refer the reader to [2] for their universal counterparts.

Path-based temporal logics with Until operators are obtained by discarding propositional variables $\mathcal{V}_F$ from the path formulas of $\mu\mathcal{L}_F$, and replacing fixpoint formulas $\eta p^F.\varphi^F$ with $\eta \in \{\mu, \nu\}$ by formulas $\varphi^F U_L \psi^F$, with $L \subseteq \Lambda_F$ a subset of (typically disjunction-preserving) predicate liftings. Furthermore, one can add negation to the syntax of both path and state formulas, and discard the requirement that only *monotone* predicate liftings should be con-

sidered in Λ and Λ_F, since no appeal to the Knaster-Tarski theorem is needed to interpret Until operators. Instead, the semantics of Until operators is defined by

$$(\!|\varphi^F U_L \psi^F|\!)_{\gamma,V} = \bigcup_{t\in\omega} (\!|\varphi^F U_L^{\leq t} \psi^F|\!)_{\gamma,V}$$

where the formulas $\varphi^F U_L^{\leq t} \psi^F$ with $t \in \omega$ are defined inductively by:

$$\begin{aligned} \varphi^F U_L^{\leq 0} \psi^F &::= \psi^F \\ \varphi^F U_L^{\leq t+1} \psi^F &::= \psi^F \vee (\varphi^F \wedge \bigvee_{\lambda^F \in L} [\lambda^F](\varphi^F U_L^{\leq t} \psi^F) \end{aligned}$$

The semantics of state formulas remains as before.

Example 4.2 One can recover the logic PCTL [7] as a fragment of the temporal logic obtained by taking $T = \mathcal{G}_1$ and $F = \mathsf{Id}$ on Meas. Predicate liftings for endofunctors $F : \mathsf{Meas} \to \mathsf{Meas}$ were considered in [15], as natural transformations of type $P \Rightarrow P \circ F$ with $P : \mathsf{Meas} \to \mathsf{Set}$ given by $P(X, \Sigma_X) = \Sigma_X$. In particular, the identity natural transformation defines a predicate lifting for $F = \mathsf{Id}$, and we write O for the associated modality. Also, for $q \in \mathbb{Q} \cap [0,1]$, the natural transformation $\lambda_q : P \Rightarrow P \circ \mathcal{G}_1$ given by $(\lambda_q)_{(X,\Sigma_X)}(Y) = \{\mu \in \mathcal{M}_1(X, \Sigma_X) \mid \mu(Y) \geq q\}$ for $Y \in \Sigma_X$ defines a predicate lifting for $T = \mathcal{G}_1$, and we write L_q for the associated modality. The logic PCTL (interpreted over measurable spaces) is now obtained by letting $\Lambda_F = \{\mathsf{O}\}$ and $\Lambda = \{L_q \mid q \in \mathbb{Q} \cap [0,1]\}$, and further simplifying the syntax of path formulas to

$$\varphi^F ::= \mathsf{O}\varphi \mid \varphi U_{\{\mathsf{O}\}} \varphi$$

Its interpretation over Markov chains with countable state spaces is then obtained by regarding each such Markov chain as a discrete measurable space. For example, the path formula $\varphi \, \mathbf{U}^{\leq \infty} \, \psi$ of PCTL is encoded as $\varphi U_{\{\mathsf{O}\}} \psi$.

Moreover, by varying the transition type to $F = \mathsf{Id}$ or $F = 1 + A \times \mathsf{Id}$, one automatically obtains variants of PCTL interpreted over generative probabilistic systems, possibly with explicit termination.

We conclude this section by noting that the full language of CTL* can be recovered using a similar approach, i.e. by defining the CTL* path operators directly rather than through fixpoint operators.

5 Concluding Remarks

We have provided a general account of possibly infinite traces and executions in systems modelled as coalgebras. The notion of infinite execution has

subsequently been used to give semantics to generic path-based coalgebraic temporal logics, instances of which subsume known path-based logics such as CTL* and PCTL. Moreover, we have shown that by simply varying the transition type, interesting variants of known logics can be obtained with very little effort.

Future work will generalise these results to arbitrary (non-affine) monads. Apart from the powerset, lift and subprobability measure monads, a non-affine monad of interest is the multiset monad, due to its relevance to graded temporal logic. The study of the relationship between finite and possibly infinite traces constitutes another direction for future work.

References

[1] C. Cîrstea and D. Pattinson. Modular construction of complete coalgebraic logics. *Theoretical Computer Science*, 388:83–108, 2007.

[2] C. Cîrstea and M. Sadrzadeh. Modular games for coalgebraic fixed point logics. In *Proc. CMCS 2008*, volume 203 of *ENTCS*, 2008.

[3] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.

[4] J. L. Doob. *Measure Theory*. Springer, 1994.

[5] E.A. Emerson and J. Halpern. "sometimes" and "not never" revisited: On branching versus linear time. *Journal of the ACM*, 33:151–178, 1986.

[6] M. Giry. A categorical approach to probability theory. In *Categorical Aspects of Topology and Analysis*, volume 915 of *Lecture Notes in Mathematics*, 1982.

[7] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[8] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3:1–36, 2007.

[9] B. Jacobs. Semantics of weakening and contraction. *Annals of Pure and Applied Logic*, 69:73–106, 1994.

[10] B. Jacobs. Trace semantics for coalgebras. In *Proc. CMCS 2004*, volume 106 of *ENTCS*, 2004.

[11] B. Jacobs and A. Sokolova. Traces, executions and schedulers, coalgebraically. In *Proc. CALCO'09*, volume 5728 of *LNCS*, 2009.

[12] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[13] D. Pattinson. Expressive logics for coalgebras via terminal sequence induction. *Notre Dame Journal of Formal Logic*, 45(1):19–33, 2004.

[14] J. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[15] C. Schubert. Coalgebraic logic over measurable spaces: behavioural and logical equivalence. Draft.

[16] C. Schubert. Final coalgebras for measure polynomial endofunctors. In *Proceedings of TAMC'09*, 2009.

[17] A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. PhD thesis, TU Eindhoven, 2005.

Applications of Algebra and Coalgebra in Scientific Modelling

Illustrated with the Logistic Map

Michael Hauhs[1] Baltasar Trancón y Widemann[2]

Ecological Modelling
University of Bayreuth
Bayreuth, Germany

Abstract

In computer science, the algebra–coalgebra duality serves as a formal framework for connecting the perspectives of state-based and behavior-based models. In other sciences such as ecology, these perspectives are seemingly harder to reconcile. We explore modelling paradigms, in the sense of philosophy of science, as an intermediate step in translating the (co)algebraic framework from computer science into applications in ecology. We illustrate the application potential of this approach with a simple model from theoretical ecology: the logistic map. Several versions of algebraic models with progressively more sophisticated carriers and operations are introduced and finally contrasted with a corresponding coalgebraic model. We illustrate two modelling paradigms with these examples. Only one of these has traditionally been used in ecology. The second one, which is based on a coalgebraic dualisation, offers new modelling perspectives in ecology and environmental science.

Keywords: algebra, coalgebra, state, behavior, model, paradigm, scientific method, dynamic system, ecology, logistic map

1 Introduction

Scientific modelling, the task of relating theories and data, is a multi-faceted problem without a single universal solution. Besides the particular discipline of science under study, it is necessarily connected to the polar areas of philosophy and mathematics.

[1] Email: Michael.Hauhs@uni-bayreuth.de
[2] Email: Baltasar.Trancon@uni-bayreuth.de

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

A fundamental dichotomy from both the philosophic and the mathematical viewpoint is the choice between *state* and *behavior* as the primary ontological category of system properties. There are some scientific disciplines where one is clearly dominant: Physical sciences tend to be state-based, whereas social sciences tend to be behavior-based. But there is also a middle ground covered by life-related sciences, in particular ecology as the science of living systems in an open environment. These sciences pose especially interesting and hard challenges to the modeller, because neither state nor behavior alone seem to be sufficient for comprehensive system descriptions.

In most scientific fields, the primacy of either state or behavior is correlated with the degree of formalization: State-based models tend to be given in mathematical formulae, whereas behavior-based models tend to be given in narrative prose. Computer science is rather distinguished by the fact that it provides methods to render both perspectives with comparable formal rigor, and to unify them in common frameworks. Of these frameworks, we regard the duality of universal algebra and coalgebra as particularly promising for scientific modelling, for several reasons:

(i) There are vast bodies of theoretical results on how to apply algebra and coalgebra to state-based (e.g. [5]) and behavior-based (e.g. [9]) system models, respectively.

(ii) The duality is a precise relationship within the meta-framework of category theory, as opposed to a mere philosophical complementarity [16].

(iii) The usefulness of commuting diagrams similar to those underlying the categorial formulation of (co)algebra for theoretical biology has already been established [15].

Our present work should be understood as a small step towards leveraging the tools of theoretical computer science for theoretical ecology. This overall goal is not easy to achieve; not least because the structural mathematics of computer science remain obscure and inaccessible to the more classically trained ecologist. As an intermediate, more modest goal, we aim at extending the repertoire of scientific ecological modelling with methods originally designed for the description of systems of logic, control and computation. Towards this end, we shall presently discuss a system that is simple and idealized, yet of some popularity in theoretical ecology. We shall illustrate that modelling questions concerning this system fall into the two aforementioned dual categories, and how they can be mapped to algebraic and coalgebraic formulations, respectively. Our focus here shall be the systematic development of modelling techniques from basic universal (co)algebra and their interpretation from the meta-viewpoint of philosophy of science; the connection to more realistic and practical ecological problems is outside the scope of this article.

1.1 Scientific Modelling

Our modelling examples will be idealized. We use the notions proposed by [6] in which modelling is composed of two steps: The first step replaces a real-world phenomenon, the *target system*, with an idealised system described in words, the *model system* (see Figure 1 ibid.). In the historical case of astronomical models of planets, the physical objects were replaced by idealised, homogenous spheres with point mass. Here we use a population of organisms and its temporal variation by growth as the target system and replace it with a spatially homogenous model system: The model system is then in the second step described by the logistic map.

A modelling paradigm, as introduced by Kuhn, links an aspect of the empirical world, the model system (in the sense above), with mathematics. It has to include a recipe, how to fill/relate the description of the word model with data on the one hand and how to symbolise the description and apply mathematics on the other hand.

Several options for this task exist. They are vastly different with respect to their reputation in science, to the extent that sometimes one paradigm, the physical one, is identified with the scientific method as such. However, empirics and management practice, at least for ecological problems, appear determined to remain methodologically diverse. We do not take sides in this dispute, and present two dual modelling paradigms without judging their relative applicability a priori.

Each of the two paradigms emphasizes one of the two ontological categories: The *functional* paradigm is based on observable states; behavior is a secondary notion that arises of the change of states under a dynamic law. The *interactive* paradigm focuses directly on behavior; state arises from the history of choices of agents. The latter paradigm is uncommon in most "hard" sciences. Again, computer science is an exception; cf. the famous Turing test.

It is here where we expect the impact of coalgebra. The new theoretical approach may formalise a model paradigm which is already implicitly used in ecological practice, but which has not been recognised in theoretical ecology [17]. A corresponding problem in the philosophy of science is the epistemological classification of computer simulations [8].

Using coalgebra to model natural phenomena is not (yet) a popular approach. This is no surprise, because few natural scientists are even aware of the existence of such a theory. The gap between the research programs of natural sciences on one hand and of theoretical computer science on the other hand makes it difficult to exchange abstract notions and theoretical frameworks. Rigorous study of scientific modelling and its integration of "mindset" and "toolkit" can be beneficial to mutual understanding.

The logistic map has been chosen as an objct of study for its simplicity,

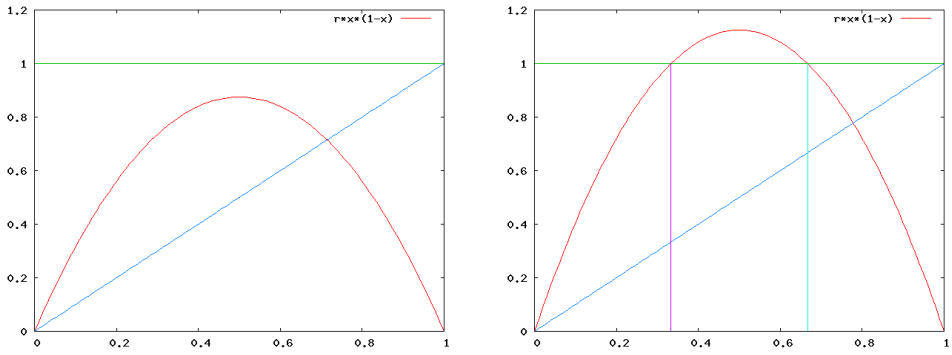


Fig. 1. The logistic map f_r for $r < 4$ (left) and $r > 4$ (right)

not for its immediate practical relevance. For a more relevant example of ecological behavior, consider the idealised case of a domesticated species in which evolutionary change can be supressed deliberately. The complete space of possible behavior *under human management* can then be derived from its documented growth history. The scientific task is to comprehensively represent patterns of this history along with proper goals and intervention norms, in order to allow a sustained continuation of the past behavior, but without being able to reconstruct the system after irreversible failure, such as extinction of the species.

1.2 The Logistic Map

The so-called logistic map [10] is related to the logistic equation published by Verhulst in 1838, which was one of founding concepts of theoretical population biology [1]. It has been criticised for being oversimplified, but is still a reference concept for more realistic models. It is used as the introductory example in a standard textbook on theoretical biology [12].

Definition 1.1 For a real parameter $r > 0$, the **logistic map** is defined as the real function

$$f_r(x) = rx(1 - x)$$

restricted in both domain and range to the unit interval $\mathbb{I} = [0, 1]$; see Figure 1.

- For $r \leq 4$, the function f_r is totally defined on $\mathbb{I}$.
- For $r > 4$, the function f_r is only partially defined on $\mathbb{I}$: $f_r(x) \notin \mathbb{I}$ for some $x \in \mathbb{I}$.

The single parameter r is interpreted as the effective growth rate of the system. The state of the system is interpreted as population density, normalised by the carrying capacity of the system with respect to the given environment.

Any relation with the environment is encoded into the carrying capacity parameter, hence this gives rise to a discrete autonomous dynamic system.

Dynamic systems with the state space $\mathbb{I}$ and the step function f_r exhibit a variety of interesting modes of behavior, depending on the value of r: from certain extinction through stable fixed points and periodic solutions of all periods to deterministic chaos with strange attractors.

The logistic map has been investigated with the methods of symbolic dynamics as an important case of a complex, chaotic system. In this role it has also been used as an application of coalgebra [16]. This, to our knowledge, has been the first connection between coalgebra and models used in biology. Here we use the well-known features of this map for reviewing the various roles in which dynamic models can be used in ecological modelling.

Time is discrete in a system with the step function f_r. This is not necessarily an idealization for biological systems; e.g. generation times. But the state space $\mathbb{I}$ is idealized as continuous. For the application of symbolic dynamics and to accomodate the realistic assumption that measurements cannot be made arbitrarily precise, we discretize observations as partitions of the state space, specified by the assignment of symbols from a finite alphabet. It suffices to consider the most coarse-grained case.

Definition 1.2 The **binary unit partition** is defined as a function $c : \mathbb{I} \to 2 = \{0, 1\}$

$$c(x) = \begin{cases} 0 & \text{if } x < \frac{1}{2} \\ 1 & \text{if } x \geq \frac{1}{2} \end{cases}$$

Note that f_r is not reversible (injective), but the tupling $\langle c, f_r \rangle : \mathbb{I} \to 2 \times \mathbb{I}$ is.

- For $r < 4$, the function $\langle c, f_r \rangle^{-1}$ is only partially defined on $2 \times \mathbb{I}$.
- For $r \geq 4$, the function $\langle c, f_r \rangle^{-1}$ is totally defined on $2 \times \mathbb{I}$.

This binary partition of the logistic dynamic system has been used in [4] to demonstrate that the apparent complexity of a system depends crucially on the viewpoint.

2 Formal Prerequisites

The mathematical structures underlying not only our example model system, but more or less directly every dynamic system, are the sequential data structures: finite and infinite sequences over a fixed set of elements. These structures and the usual ways of reasoning with them have well-understood representations in terms of algebra and coalgebra.

2.1 Strings and Streams

Definition 2.1 The set A^* is called the set of **finite sequences** or **strings** over A. It is generated by the free constructors $\mathrm{cons}_A : A \times A^* \to A^*$ and $\mathrm{nil}_A \in A^*$. The destructors are the unique partial functions $\mathrm{hd}_A : A^* \nrightarrow A$ and $\mathrm{tl}_A : A^* \nrightarrow A^*$ such that

$$\begin{aligned} \mathrm{hd}_A\big(\mathrm{cons}_A(a,w)\big) &= a & \qquad \mathrm{hd}_A(\mathrm{nil}_A) &\text{ undefined} \\ \mathrm{tl}_A\big(\mathrm{cons}_A(a,w)\big) &= w & \qquad \mathrm{tl}_A(\mathrm{nil}_A) &\text{ undefined} \end{aligned}$$

- We omit all subscript annotations where no ambiguity arises.
- We define the subsets $A^n \subset A^*$ of strings of a fixed length n inductively as

  $$A^{n+1} = \mathcal{P}(\mathrm{cons}_A)(A \times A^n) \qquad A^0 = \{\mathrm{nil}_A\}$$

  Then

  $$A^* = \bigcup_{n=0}^{\infty} A^n \qquad A^+ = \bigcup_{n=1}^{\infty} A^n = A^* \setminus \{\mathrm{nil}_A\}$$

- We informally write $a_1 \ldots a_n$ for $\mathrm{cons}(a_1, \ldots, \mathrm{cons}(a_n, \mathrm{nil}) \ldots)$.
- In particular, we abbreviate a singleton string $\mathrm{cons}(a, \mathrm{nil})$ to a.
- For any function $f : A \to B$, we write $f^* : A^* \to B^*$ for the elementwise mapping

  $$f^*\big(\mathrm{cons}_A(a,w)\big) = \mathrm{cons}_B\big(f(a), f^*(w)\big) \qquad f^*(\mathrm{nil}_A) = \mathrm{nil}_B$$

  This turns * into a functor.

The choice of cons and nil as the constructors of strings suggest that the organization of data in a string obey the *stack* principle: data elements are accumulated and removed at the left end of a string only. The following auxiliary function handles a special case of this principle, namely the accumulation of data arising from the iterated application of a given function.

Definition 2.2 Let A be any set and $f : A \nrightarrow A$ a partial function. The partial function $\mathrm{push}(f) : A^* \nrightarrow A^*$ is defined as

$$\mathrm{push}(f)(w) = \mathrm{cons}\Big(f\big(\mathrm{hd}(w)\big), w\Big)$$

- *Strict* application is implied: $\mathrm{push}(f)$ is undefined at w if $\mathrm{hd}(w)$ or $f\big(\mathrm{hd}(w)\big)$ is undefined. In particular, $\mathrm{push}(f)(\mathrm{nil})$ is never defined.

- Note that $\mathrm{push}(f)^n : A^m \to A^{m+n}$ for $m > 0$ and $n \geq 0$. In slight abuse of notation we define $\mathrm{push}(f)^{-n} : A^m \to A^{m-n}$ for $m \geq n \geq 0$ as the retraction

$$\mathrm{push}(f)^{-n} = \mathrm{tl}^n$$

Example 2.3 The expression $\mathrm{push}(\mathrm{succ})^n(0)$ yields a countdown from n.

Definition 2.4 The set $A^\omega = (\mathbb{N} \to A)$ is called the set of **infinite sequences** or **streams** over A. Its elements are of the form $\mathrm{cons}_A(a, s)$ for any $a \in A; s \in A^\omega$, with

$$\mathrm{cons}_A(a,s)(n) = \begin{cases} a & \text{if } n = 0 \\ s(n-1) & \text{if } n > 0 \end{cases}$$

We write

$$\mathrm{hd}_A(s) = s(0) \qquad\qquad \mathrm{tl}_A(s) = s \circ \mathrm{succ}$$

for the total destructors.

2.2 (Co)Algebras of Affine Type

Definition 2.5 The family of **affine** functors $\mathcal{A}^A_B : \mathbf{Set} \to \mathbf{Set}$ is defined as

$$\mathcal{A}^A_B(X) = A \times X + B \qquad\qquad \mathcal{A}^A_B(f) = \mathrm{id}_A \times f + \mathrm{id}_B$$

We write

$$\mathrm{go}_A : A \times X \to \mathcal{A}^A_B(X) \qquad\qquad \mathrm{stop}_B : B \to \mathcal{A}^A_B(X)$$

for the left and right injection, respectively.

Lemma 2.6 *Affine functors have initial algebras. The structure* $\big(A^* \times B, \alpha = [\alpha_1, \alpha_2]\big)$ *with*

$$\alpha_1\big(a, (w, b)\big) = \big(\mathrm{cons}(a, w), b\big) \qquad\qquad \alpha_2(b) = (\mathrm{nil}, b)$$

is an initial $\mathcal{A}^A_B$*-algebra. The unique homomorphism or* ***catamorphism*** h *into any* $\mathcal{A}^A_B$*-algebra* $\big(C, \gamma = [\gamma_1, \gamma_2]\big)$ *is defined recursively as*

$$h\big(\mathrm{cons}(a, w), b\big) = \gamma_1\big(a, h(w, b)\big) \qquad\qquad h(\mathrm{nil}, b) = \gamma_2(b)$$

Lemma 2.7 *Affine functors have final coalgebras. The structure* $\big((A^* \times B) + A^\omega, \phi = [\phi_1, \phi_2]\big)$ *with*

$$\phi_1\big(\mathrm{cons}(a, w), b\big) = \mathrm{go}\big(a, \iota_1(w, b)\big) \qquad \phi_2(s) = \mathrm{go}\big(\mathrm{hd}(s), \iota_2(\mathrm{tl}(s))\big)$$
$$\phi_1(\mathrm{nil}, b) = \mathrm{stop}(b)$$

where ι_1, ι_2 are the injections into $(A^* \times B) + A^\omega$ is a final $\mathcal{A}_B^A$-coalgebra. The unique homomorphism or ***anamorphism*** *h from any $\mathcal{A}_B^A$-coalgebra (C, γ) is defined corecursively as*

$$\phi\big(h(c)\big) = \begin{cases} \mathrm{go}\big(a, h(c')\big) & \text{if } \gamma(c) = \mathrm{go}(a, c') \\ \mathrm{stop}(b) & \text{if } \gamma(c) = \mathrm{stop}(b) \end{cases}$$

Instantiating A or B with the empty set or the singleton set $1 = \{\star\}$ yields cases of special interest.

(i) The affine functor $\mathcal{A}_B^1$. The operations of $\mathcal{A}_B^1$-algebras are of type $\gamma : 1 \times C + B \to C$. They are in natural one-to-one correspondence to pairs (f, g) of type $f : C \to C$ and $g : B \to C$, namely $(f, g) \leftrightarrow [f \circ \pi_2, g]$. The carrier of the canonical initial $\mathcal{A}_B^1$-algebra (Lemma 2.6) simplifies to $\mathbb{N} \times B$ by reading 1^* as a unary number system. Its operation is specified by (f_0, g_0) with $f_0(n, b) = (n+1, b)$ and $g_0(b) = (0, b)$. The catamorphism i into the $\mathcal{A}_B^1$-algebra specified by (f, g) is the *iteration* operator

$$i(n, b) = f^n\big(g(b)\big)$$

(ii) The affine functor $\mathcal{A}_1^A$. The operations of $\mathcal{A}_1^A$-algebras are of type $\gamma : A \times C + 1 \to C$. They are in natural one-to-one correspondence to pairs (f, e) of type $f : A \times C \to C$ and $e \in C$, namely $(f, e) \leftrightarrow [f, \hat{e}]$ where $\hat{e}(*) = e$. The carrier of the canonical initial $\mathcal{A}_1^A$-algebra simplifies to A^*. Its operation is specified by $(\mathrm{cons}_A, \mathrm{nil}_A)$. The catamorphism j into the $\mathcal{A}_1^A$-algebra specified by (f, e) is the *fold* operator

$$j\big(\mathrm{cons}(a, w)\big) = f\big(a, j(w)\big) \qquad\qquad j(\mathrm{nil}) = e$$

(iii) The affine functor $\mathcal{A}_\varnothing^A$. The initial $\mathcal{A}_\varnothing^A$-algebra is empty. The operations of $\mathcal{A}_\varnothing^A$-coalgebras are of type $\gamma : C \to A \times C + \varnothing$. They are in natural one-to-one correspondence to pairs (h, t) of type $h : C \to A$ and $t : C \to C$, namely $(h, t) \leftrightarrow \iota_1 \circ \langle h, t \rangle$. The carrier of the canonical final $\mathcal{A}_\varnothing^A$-coalgebra (Lemma 2.7) simplifies to A^ω. Its operation is specified by $(\mathrm{hd}_A, \mathrm{tl}_A)$. The anamorphism k from the $\mathcal{A}_\varnothing^A$-coalgebra specified by (h, t) is the *unfold* operator

$$k(c) = \mathrm{cons}\Big(h(c), k\big(t(c)\big)\Big)$$

(iv) The affine functor $\mathcal{A}_B^\varnothing$ is degenerate and equivalent to the constant functor B.

We shall demonstrate that each nondegenerate case corresponds to a scientific modelling scenario. We use the pair notation of the preceding paragraphs to specify operations, in order to avoid cluttering diagrams with uninformative projections and injections.

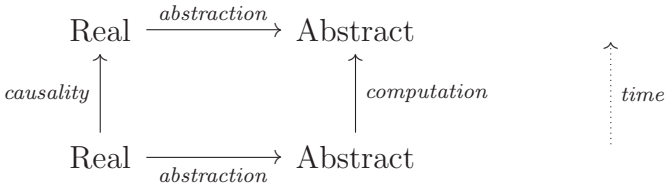


Fig. 2. Functional modelling, conceptually

3 Modelling Paradigms and (Co)Algebra

The trajectories (time-indexed sets of contiguous states) of a dynamic system have been termed *recursive* by Rosen [15], but not in the rigorous sense of theoretical computer science. For discrete-time systems, where trajectories are sequences, the metaphor can be made precise by connecting finite/infinite trajectories with iteration/coiteration in the form of catamorphisms/anamorphisms, respectively. In this section, we discuss the transition from a philosophical view on modelling paradigms to formal systems that employ initial algebras and catamorphisms or final coalgebras and anamorphisms, respectively.

3.1 From Functional Modelling to Algebra

Our proposed mapping of the two modelling paradigms to (co)algebra is inspired by [15], where the functional paradigm is discussed in great philosophical detail and organized in the form of the commuting diagram depicted in Figure 2. We have adapted the original discussion to ecological problems in [8]. Note that the *real* side refers to the model system, not the target system. We identify the situation in this diagram with a pair of algebras, namely the real and the abstract one, with states as their elements, and the abstraction with a homomorphism. A model consists of

(i) the abstraction mapping that separates essential from accidental properties of real objects, and

(ii) a logical theory (system of equations) that specifies the valid progressions of abstract states.

The abstract algebra is merely a mathematical implementation of the specification. A scientific hypothesis is posed by claiming that the diagram commutes. Unlike in pure mathematics, and in the face of uncertainty about the model system, this is not a logical property to be decied, but rather an empirical property to be judged by testing and evidence, as prescribed by the Scientific Method. If the correspondence between the two sides actually holds, it gives clauses of the specification the special status of *laws of nature*. Reverting the top horizontal arrow results in the standard test situation for functional models, the *prediction*.

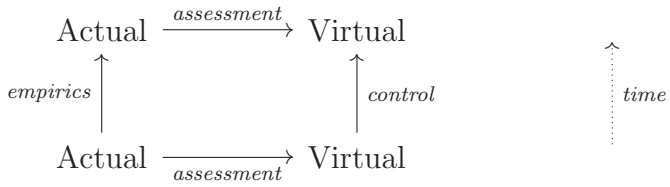


Fig. 3. Interactive modelling, conceptually

These philosophical interpretations need to be both formalized and generalized in order to adequately capture the tasks and capabilities of the functional modelling paradigm. We call algebraic modelling in the above, narow sense *direct* and distinguish it from *inverse* problems where not future states, but past or boundary conditions are investigated.

The basic tenet of algebraic modelling of both directions is to employ an initial algebra of a suitable functor as a formal *query language*, arbitrary algebras of the same functor (models) as implementations of query constructs and the catamorphisms as the recursive evaluation of queries.

3.2 From Interactive Modelling to Coalgebra

Changing the perspective from state to behavior affects all parts of the modelling situation. States are no longer required to be observable, but may be largely hidden behind an *interface*; all relevant information is taken from behavior at the interface. Metaphysically, objects and their properties are replaced by subjects and their actions. We reflect this shift of perspective, as common in the field of philosophy of science, by distinguishing the terms *real* (literally: of the things) and *actual* (literally: of the actions). Laws are replaced by their subjective counterparts, such as *strategies* and *norms*. Figure 3 shows the resulting commuting diagram. The standard test situation for interactive models is obtained by reversing the right vertical arrow; it describes *planning*. See Section 4.3 for a derivation of this model paradigm from formal representations.

The claim that coalgebraic modelling departs from the state-based perspective may be surprising. This issue arises from a fundamental difference between the notions of state in physics and in computer science. The observed state of a physical system *is* objective reality. The state of a formal automaton, as opposed to its physical implementation, merely *refers to* its actual behavior, in the sense that semantics are given in terms of observed transitions not states; for instance as the regular language accepted by a finite automaton. The reference character of state is expressed formally by the notion of bisimulation between alternative virtual systems or by final coalgebraic semantics. We conjecture that this reflects the empirical phenomenon of *equifinality* [18,2]: The observed behavior of a complex system at a simpler interface can often be reconstructed by many different processes within the

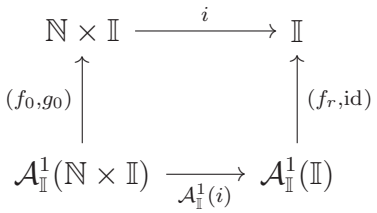


Fig. 4. Direct functional modelling (perfect information) with initial algebra

functional paradigm.

The basic tenet of coalgebraic modelling, in our sense, is to employ a final coalgebra as a formal *semantic domain*, arbitrary coalgebras of the same functor (models) as representatives of behavior and the anamorphisms as the recursive assessment of the represented behavior. The distinction between direct and inverse problems of coalgebraic modelling is less pronounced than in the algebraic case, at least for the example of the logistic map.

4 Formal Modelling Scenarios

4.1 Direct Functional Modelling

Direct functional modelling is a scenario where the "true" dynamics of a system are known. It solves the problem of *prediction*: From the observation of a current system state, future states are derived by formal (automatic) reasoning.

Claim 4.1 *The initial algebra of the functor $\mathcal{A}^1_B$, where B is the representation of system states, is paradigmatic for direct functional modelling.*

4.1.1 Perfect Information

The simplest case of direct functional modelling assumes perfect information about the precise current system state. Its application to the logistic map is shown in Figure 4. (Recall that the labels of vertical arrows are shorthands as defined in section 2.2.) The state space $\mathbb{I}$ is represented one-to-one. The left hand side is the simplified canonical initial $\mathcal{A}^1_{\mathbb{I}}$-algebra. The right hand side is a $\mathcal{A}^1_{\mathbb{I}}$-algebra that encodes the known dynamics of the system: Its carrier is the state space $\mathbb{I}$ and its operation is specified by the step function f_r (with $\mathrm{id}_{\mathbb{I}}$ as the trivial base case).

Theorem 4.2 *The catamorphism i for the operation specified by $(f_r, \mathrm{id}_{\mathbb{I}})$ solves the problem of predicting a state n steps in the future, for $r \leq 4$.*

$$i(n, x) = f_r^n(x)$$

The preceding scenario is a straightforward reconstruction of the iterated

step function f_r^n. The graph of the function consists of pairs of initial and final states, n steps apart; the intermediate states are forgotten. This can be remediated by a simple refinement that replaces single states with stack-based representations of trajectories.

Definition 4.3 We define the set of **partial trajectories** as the set of stacks (strings constructed right-to-left) arising by iterated action of f_r on any initial state ($\mathcal{P}$ is the image functor).

$$T_r = \bigcup_{n=0}^{\infty} \mathcal{P}\big(\mathrm{push}(f_r)\big)^n(\mathbb{I}^1) \subseteq \mathbb{I}^+$$

- This is the smallest set such that $\mathbb{I}^1 \subseteq T_r$ and $\mathrm{push}(f_r) : T_r \to T_r$.

The refined model is shown in Figure 5. The carrier of the right hand side algebra is changed to T_r, and the operations f_r and $\mathrm{id}_{\mathbb{I}}$ have been replaced by $\mathrm{push}(f_r)$ and $\mathrm{in}_{\mathbb{I}}$, respectively, where $\mathrm{in}_A : A \to A^1$ is the injection of singleton strings.

Theorem 4.4 *The catamorphism i for the operation specified by $\big(\mathrm{push}(f_r), \mathrm{in}_{\mathbb{I}}\big)$ solves the problem of predicting all states up to n steps in the future, for $r \leq 4$.*

$$i(n, x) = \mathrm{push}(f_r)^n(x)$$

The following three cases refine the representation of state and dynamics by replacing the state space $\mathbb{I}$ with progressively more complicated, derived spaces and replacing the step function f_r with an appropriate lifting to the respective space. Note that the requirement $r \leq 4$ is lifted.

4.1.2 Imperfect Information: Nondeterminism

A moderately simple case of direct functional modelling with imperfect information assumes nondeterminism. Note that the term "nondeterminism" is used in the usual sense of computer science, replacing the single precise current system state by a set of potential current system states. It is not used in the sense of philosophy, namely that a hidden variable, external source of

$$\begin{array}{ccc} \mathbb{N} \times \mathbb{I} & \xrightarrow{\quad i \quad} & T_r \\ {\scriptstyle (f_0, g_0)}\uparrow & & \uparrow{\scriptstyle (\mathrm{push}(f_r), \mathrm{in})} \\ \mathcal{A}_{\mathbb{I}}^1(\mathbb{N} \times \mathbb{I}) & \xrightarrow[\mathcal{A}_{\mathbb{I}}^1(i)]{} & \mathcal{A}_{\mathbb{I}}^1(T_r) \end{array}$$

Fig. 5. Direct functional modelling (partial trajectories) with initial algebra

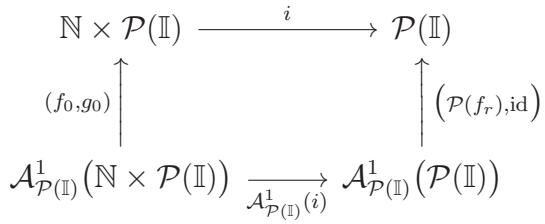


Fig. 6. Direct functional modelling (nondeterministic) with initial algebra

randomness or decision-making entity is involved in the transition from one state to another.

The application of nondeterminstic direct functional modelling to the logistic map is shown in Figure 6. The state space $\mathbb{I}$ is represented by its powerset $\mathcal{P}(\mathbb{I})$. The left hand side is the simplified canonical initial $\mathcal{A}^1_{\mathcal{P}(\mathbb{I})}$-algebra. The right hand side is an $\mathcal{A}^1_{\mathcal{P}(\mathbb{I})}$-algebra that encodes the nondeterministic dynamics of the system: Its carrier is the set $\mathcal{P}(\mathbb{I})$ of sets of potential states and its operation is specified by $\mathcal{P}(f_r)$, the image of state sets under f_r; a state is a potential post-state of a step if and only if it is the image of a potential pre-state under f_r.

Theorem 4.5 *The catamorphism i for the operation specified by $\big(\mathcal{P}(f_r), \mathrm{id}_{\mathcal{P}(\mathbb{I})}\big)$ solves the problem of predicting a nondeterministic state n steps in the future.*

$$i(n, Y) = \mathcal{P}(f_r)^n(Y)$$

The nondeterministic case can be extended to more sophisticated imperfect information such as fuzzy sets of potential states.

4.1.3 Imperfect Information: Probabilism

Definition 4.6 Each continuous probability distribution on $\mathbb{I}$ is specified uniquely by a **cumulative distribution function (cdf)**, that is a continuous, weakly mononotic function $F : \mathbb{I} \to \mathbb{I}$ with $F(0) = 0$ and $F(1) = 1$.

- An $\mathbb{I}$-valued random variable X is said to be distributed according to F, written $X \sim F$, if and only if $F(y) = P(X \leq y) = P(X < y)$.
- We write $\widetilde{\mathbb{I}}$ for the set of cdfs on $\mathbb{I}$.

Definition 4.7 The function $\widetilde{f_r} : \widetilde{\mathbb{I}} \to \widetilde{\mathbb{I}}$ is defined as

$$\widetilde{f_r}(F)(y) = F\big(\tfrac{1}{2} - q_r(y)\big) + 1 - F\big(\tfrac{1}{2} + q_r(y)\big) \quad q_r(y) = \begin{cases} \sqrt{\frac{1}{4} - \frac{y}{r}} & \text{if } y \leq \frac{r}{4} \\ 0 & \text{if } y > \frac{r}{4} \end{cases}$$

It is easy to verify that $\widetilde{f_r}(F)$ is in fact a cdf on $\mathbb{I}$. Note that $\frac{1}{2} \pm q_r(y)$ is the position of the vertical markers in Figure 1, right hand side.

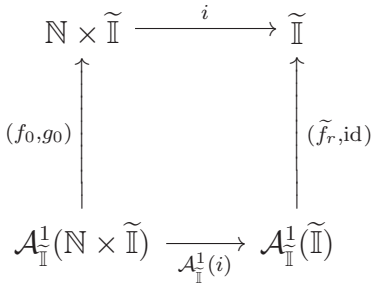


Fig. 7. Direct functional modelling (probabilistic) with initial algebra

Lemma 4.8 *The function $\widetilde{f}_r$ lifts a distribution over the function f_r.*

$$X \sim F \implies f_r(X) \sim \widetilde{f}_r(F)$$

Proof.

$$\begin{aligned}
P\big(f_r(X) \le y\big) &= P\big(rx(1-x) \le y\big) \\
&= P\Big(\big(x - \tfrac{1}{2}\big)^2 \ge \tfrac{1}{4} - \tfrac{y}{r}\Big) \\
&= P\Big(\big|x - \tfrac{1}{2}\big| \ge q_r(y)\Big) \\
&= P\big(x \le \tfrac{1}{2} - q_r(y) \vee x \ge \tfrac{1}{2} + q_r(y)\big) \\
&= P\big(x \le \tfrac{1}{2} - q_r(y)\big) + P\big(x \ge \tfrac{1}{2} + q_r(y)\big) \\
&= P\big(x \le \tfrac{1}{2} - q_r(y)\big) + 1 - P\big(x \le \tfrac{1}{2} + q_r(y)\big) \\
&= F\big(\tfrac{1}{2} - q_r(y)\big) + 1 - F\big(\tfrac{1}{2} + q_r(y)\big) \\
&= \widetilde{f}_r(F)(y)
\end{aligned}$$

□

The application of probabilistic direct functional modelling to the logistic map is shown in Figure 7. The state space $\mathbb{I}$ is represented by the set of cdfs $\widetilde{\mathbb{I}}$. The left hand side is the simplified canonical initial $\mathcal{A}^1_{\widetilde{\mathbb{I}}}$-algebra. The right hand side is a $\mathcal{A}^1_{\widetilde{\mathbb{I}}}$-algebra that encodes the probabilistic dynamics of the system: Its carrier is the set $\widetilde{\mathbb{I}}$ of state distributions and its operation is specified by $\widetilde{f}_r$, the action of f_r on the distribution of its argument.

Theorem 4.9 *The catamorphism i for the operation specified by $(\widetilde{f}_r, \mathrm{id}_{\widetilde{\mathbb{I}}})$ solves the problem of predicting a probabilistic state n steps in the future.*

$$X \sim F \implies f_r^n(X) \sim i(n, F)$$

The probabilistic case can be extended to more complex, not purely continuous distributions.

4.2 Inverse Functional Modelling

Inverse functional modelling is a scenario where inferences about the dynamics of a system (parameters, initial or boundary conditions) are drawn from data recorded by external observation. It solves the problem of *reconstruction*: Empirical observations are reduced to possible causes (parameters and conditions not directly observable, but consistent with the data).

Claim 4.10 *The initial algebra of the functor $\mathcal{A}_1^A$, where A is the range of the observable system property of interest, is paradigmatic for inverse functional modelling.*

We choose the binary partition c as observable property. Its range is the binary alphabet 2, hence the carrier of the canonical initial algebra is the language of binary strings 2^*.

Definition 4.11 The function $w_r : \mathbb{N} \times \mathbb{I} \to 2^*$ is defined as

$$w_r(n, x) = c^*\big(\mathrm{push}(f_r)^{n-1}(x)\big)$$

It maps the pair (n, x) to the stack of observed binary symbols for n consecutive system states starting with x. Informally,

$$w_r(n, x) = c\big(f_r^{n-1}(x)\big) \cdots c\big(f_r^0(x)\big)$$

- The range of w_r for $n > 0$ is the set of partitioned partial trajectories $\mathcal{P}(c^*)(T_r)$.

The inverse modelling task, given data $w \in 2^*$ of length n and a parameter value r, is to find some or all solutions of the equation $w = w_r(n, x)$. A concise representation of the inferred information is given by a partial function on $\mathbb{I}$ that is defined only for initial states consistent with the observed data, and maps those to the final states after the observation. The solution is straightforwardly constructed, dealing with one observed symbol at a time.

Definition 4.12 We write $\overrightarrow{\mathbb{I}} = (\mathbb{I} \rightharpoonup \mathbb{I})$ for the space of partial functions on $\mathbb{I}$. The function $\overrightarrow{f_r} : 2 \times \overrightarrow{\mathbb{I}} \to \overrightarrow{\mathbb{I}}$ is defined as

$$\overrightarrow{f_r}(a, h) = f_r|_a \circ h \qquad \text{where} \quad f_r|_a = \begin{cases} f_r(x) & \text{if } c(x) = a \\ \text{undefined} & \text{if } c(x) \neq a \end{cases}$$

The operation $\overrightarrow{f_r}$ refines and extends a given partial function h by excluding initial states that are mapped by h to intermediate states inconsistent with a given data symbol a, and taking all others one f_r-step further.

The application of inverse functional modelling to the logistic map is shown in Figure 8. The observation range is the binary alphabet 2. The left hand

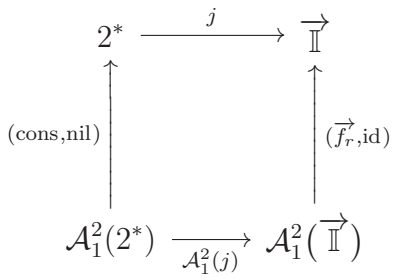


Fig. 8. Inverse functional modelling with initial algebra

side is the simplified canonical $\mathcal{A}_1^2$-algebra. The right hand side is a $\mathcal{A}_1^2$-algebra that encodes the elementwise refinement of inference: Its carrier, the "state space" of inference, is the set of partial functions $\overrightarrow{\mathbb{I}}$. Its operation is specified by $\overrightarrow{f_r}$, the action of f_r on the inference for its argument.

Theorem 4.13 *The catamorphism j for the operation specified by $(\overrightarrow{f_r}, \mathrm{id}_{\overrightarrow{\mathbb{I}}})$ solves the problem of inferring initial conditions from finite data.*

$$j(w)(x) = \begin{cases} f_r^n(x) & \text{if } w = w_r(n,x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Corollary 4.14 *The domain of $j(w)$ is a sound and monotonic approximate reconstruction of the initial state from finite data w, analogous to the method of nested intervals: Let $Y_r(n,x) = \mathrm{dom}\Big(j\big(w_r(n,x)\big)\Big)$. Then for all $m, n \geq 0$*

$$x \in Y_r(n,x) \qquad\qquad m < n \implies Y_r(m,x) \supseteq Y_r(n,x)$$

Here in the context of algebraic modelling, chronicles of events (observations of behavior) are used as means for identifying the initial and final state or the dynamics of the system under study. In the equation

$$j(a_1 \ldots a_n) = f_r|_{a_1} \circ \cdots \circ f_r|_{a_n}$$

however, an alternative view becomes apparent: The standard technique of category theory is to study objects without reference to their internal structure by studying the external structure of morphisms around them. Applied to the model above, this means studying the set of chronicles without reasoning about points of the transition functions they describe via j. This allows us to consider the limit $n \to \infty$, and represent complete, infinite behavior, for which the interpretation as end-to-end transition functions breaks down. This step takes us to the interactive paradigm on the philosophical level, and to coalgebra on the mathematical level.

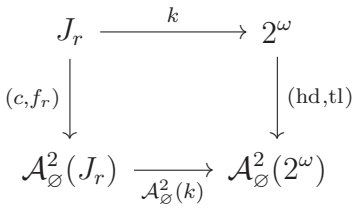


Fig. 9. Direct interactive modelling with final coalgebra

4.3 Interactive Modelling

Interactive modelling is a scenario where the observable properties of a system are represented without referring to any particular process as their cause. It solves the problem of *assessment*: System states are no longer observed directly but classified according to their potential (future) behavior.

Claim 4.15 *The final coalgebra of the functor $\mathcal{A}^A_\varnothing$, where A is the range of the observable system property of interest, is paradigmatic for interactive modelling.*

For $r > 4$, the logistic map is not bounded by the interval $\mathbb{I}$; we treat the case that the interval is exceeded as undefined. The set $\mathrm{dom}(f_r)$ of points for which a single step is defined is easily characterized, but the set of points for which unboundedly many steps are defined is nontrivial. The following characterization and model are derived from [16].

Definition 4.16 The set J_r such that $f_r^n(x) \in \mathbb{I}$ for all $x \in J_r$ and $n > 0$ is ($\overline{\mathcal{P}}$ is the preimage functor)

$$J_r = \bigcap_{n=0}^{\infty} \overline{\mathcal{P}}(f_r)^n(\mathbb{I}) \subseteq \mathbb{I}$$

- This is the largest set such that $J_r \subseteq \mathbb{I}$ and $f_r : J_r \to J_r$.
- If $r \leq 4$ then $J_r = \mathbb{I}$; otherwise J_r is a complicated (fractal) subset of $\mathbb{I}$.
- Note the duality to T_r in Definition 4.3.

Lemma 4.17 *For $r > 4$, the structure (J_r, γ) where γ is specified by (c, f_r) is a final $\mathcal{A}^2_\varnothing$-coalgebra.*

Proof. Section 18 of [16] gives an isomorphism $\tilde{c}$ between certain coalgebras over the category of complete metric spaces. Forgetting the metric structure, the following equations remain.

$$\begin{aligned} \mathrm{hd} \circ \tilde{c} &= c & \mathrm{tl} \circ \tilde{c} &= \tilde{c} \circ f_r \\ &= \mathrm{id}_2 \circ c \end{aligned}$$

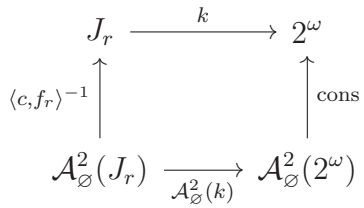


Fig. 10. Inverse interactive modelling with final coalgebra

Simple calculation yields

$$\underbrace{\iota_1 \circ \langle \mathrm{hd}, \mathrm{tl} \rangle}_{(\mathrm{hd},\mathrm{tl})} \circ \tilde{c} = \big((\mathrm{id}_2 \times \tilde{c}) + \mathrm{id}_\varnothing\big) \circ \iota_1 \circ \langle c, f_r \rangle$$
$$= \mathcal{A}^2_\varnothing(\tilde{c}) \circ \underbrace{\iota_1 \circ \langle c, f_r \rangle}_{(c,f_r)}$$

That is, $\tilde{c}$ is a homomorphism, and hence isomorphism, between the coalgebras depicted in Figure 9. Since the right hand side is final, the left hand side is also final, and the isomorphism is the anamorphism k. □

Theorem 4.18 *The anamorphism k for the operation specified by (c, f_r) solves the problem of representing the complete future behavior at the interface defined by c. Representations of the form $k(x)$ do not contain any reference to the parameter r or the initial value x.*

$$c\big(f_r^n(x)\big) = \mathrm{hd}\Big(\mathrm{tl}^n\big(k(x)\big)\Big) = k(x)(n)$$

This representation allows complete, infinite trajectories to be specified in the form $k^{-1}(s)$, in terms of a binary stream $s \in 2^\omega$. Empirical, finite data of behavior at the interface, formally collected using $w_r(n, x)$, is generally not sufficient to specify a trajectory uniquely in this way—an instance of the epistemological problem of induction; there is no logically safe procedure for obtaining nontrivial universal empirical truths [7,14]. This leads to a dual of the problem of measurement precision in state-based modelling, namely the problem of *complete* chronicles of behavior. A collection of data is complete in this sense if extrapolation from the observed strings to the possible streams is safe under given boundary conditions.

We have noted in Definition 1.2 that the operation $\langle c, f_r \rangle$ is bijective on both $\mathbb{I}$ and the subset J_r for $r \geq 4$. Incidentally, the latter is the operation of the $\mathcal{A}^2_1$-coalgebra depicted on the left hand side of Figure 9. Since the operation of the final $\mathcal{A}^2_1$-coalgebra is also bijective (by Lambek's Lemma), we may reverse the vertical arrows to arrive at the diagram shown in Figure 10. Note that the distinction between algebra and coalgebra is rather blurred in this scenario.

The operation $\langle c, f_r \rangle^{-1} : 2 \times J_r \to J_r$ models a non-autonomous dynamic system with binary input in each step. This input may be interpreted as the nondeterministic *choice* of an agent, either internal or external to the system. Under this interpretation, prediction is no longer a valid problem. But this apparent restriction is actually a trade-off: On the upside, it becomes possible to investigate actually observed, contingent behavior in virtualized form in terms of subcoalgebras of the final coalgebra. Laws regarding the presence or absence of certain patterns in these subsystems, described by a theory in modal logic, reflect *strategies* in the actual system, the dual of natural laws. Examples of the relevance of strategies, both literally and figuratively, abound in ecology: Organisms prefer favourable and avoid hostile environments, natural selection is most effectively described in strategic terms, ecosystem use is governed by economic rationale and social norms; cf. the domestication example in section 1.1.

5 Conclusion

We have demonstrated that for the simple logistic model, the relationship between the functional and the interactive modelling paradigm can be made formally precise as the algebra–coalgebra dualism. Since dualism is not equivalence, the key issue for further research is where the two approaches deviate, both on the empirical level regarding the role of data and on the theoretical level regarding the role of formalisms. The keywords of both paradigms are given in Table 1 in synopsis.

In ecology and environmental sciences, the functional paradigm is prevalent but not unconditionally successful [13]. Therefore, the added value of interactive models is of particular interest. Many essential features of living systems are naturally characterized in terms of behavior, e.g. feeding, reproducing, growing, evolving. Being alive is not a state property in the functional sense, as the development towards artificial life has shown [3]. Coalgebraic modelling facilitates the formal organization of chronicles, as opposed to measurements; this may prove an important extension in this context.

Interactive theories formulated in coalgebra not only have a different formal presentation, they encode different pragmatics. On the functional side, problems of prediction and reconstruction are solved by searching for laws that govern the dynamic mechanism. On the interactive side, problems of assessment and management are solved by searching for strategies, norms or intentions that govern the behavior of agents. The transition from the former to the latter paradigm will not solve the notoriously difficult problems about explaining ecosystems, but offers the opportunity to formalize models of sustaining ecosystems.

Table 1
Modelling paradigms and keywords

Paradigm	**Functional**	**Interactive**
Ontological Basis	state	behavior
Origin of Formalisms/Metaphors	physics	computer science
Empirical Reference	simple building blocks w. invariants	complete behavior history w. utilization record
Tests	prediction reconstruction	assessment planning
Mathematic Structure	algebra	coalgebra
Logic Theory Example	equational energy conservation	modal sustainable use
Application Domains	geosciences weather forecast	simulation, games ecosystem management

5.1 Related Work

The inspiration to use the logistic map to demonstrate the potential of the algebra–coalgebra duality for scientific modelling has been taken from [16], where the result that forms the foundation of our interactive modelling scenario is given rather in passing.

The characterization of functional and interactive modelling as commutative diagrams has been given in [17], where we have criticised the situation of theoretical ecology from the perspective of software science.

The technique of realizing (co)recursive operations as cata-/anamorphisms of simpler operations has been adapted from the *Squiggol* approach to constructive functional programming; confer the famous banana notation from [11].

References

[1] Berryman, A., *The origins and evolution of predator–prey theory*, Ecology **73** (1992), pp. 1530–1535.

[2] Beven, K. J., *A manifesto for the equifinality thesis*, Journal of Hydrology **320** (2006), pp. 18–36.

[3] Brooks, R., *The relationship between matter and life*, Nature **409** (2001), pp. 409–411.

[4] Crutchfield, J. P., *Observing complexity and the complexity of observation*, in: H. A. Atmanspacher and G. J. Dalenoort, editors, *Inside versus Outside*, Springer Series in Synergetics, Springer-Verlag, Berlin, 1994 pp. 235–272.

[5] Ehrig, H. and B. Mahr, "Fundamentals of Algebraic Specification I. Equations and Initial Semantics," Springer-Verlag, Berlin, 1985.

[6] Frigg, R., *Fiction and scientific representation*, in: R. Frigg and M. Hunter, editors, *Beyond Mimesis and Nominalism: Representation in Art and Science*, Springer-Verlag, Berlin, 2009 .

[7] Goodman, N., "Fact, Fiction, & Forecast," Harvard University Press, 1955.

[8] Hauhs, M. and H. Lange, *Foundations for the simulation of ecosystems*, in: J. Lenhard, G. Küppers and T. Shinn, editors, *Simulation: Pragmatic Constructions of Reality*, number 25 in Sociology of the Sciences Yearbook, Kluwer Academic Publishers, Dordrecht, 2006 pp. 57–77.

[9] Jacobs, B., *Exercises in coalgebraic specification*, in: *Algebraic and coalgebraic methods in the mathematics of program construction*, Springer-Verlag, New York, 2002 pp. 237–280.

[10] May, R. M., *Simple mathematical models with very complicated dynamics*, Nature **261** (1976), pp. 459–467.

[11] Meijer, E., M. Fokkinga and R. Paterson, *Functional programming with bananas, lenses, envelopes and barbed wire*, in: *Proceedings of the 5th ACM conference on Functional programming languages and computer architecture* (1991), pp. 124–144.

[12] Murray, J. D., "Mathematical Biology," Springer-Verlag, Heidelberg, 1989.

[13] Peters, R. H., "A Critique for Ecology," Cambridge University Press, 1991.

[14] Popper, K. R. and D. W. Miller, *A proof of the impossibility of inductive probability*, Nature **302** (1983), pp. 687–688.

[15] Rosen, R., "Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life," Columbia University Press, New York, 1991.

[16] Rutten, J. J. M. M., *Universal coalgebra: a theory of systems*, Theor. Comput. Sci. **249** (2000), pp. 3–80.

[17] Trancón y Widemann, B. and M. Hauhs, *Programming as a model for the theory of ecosystems*, in: J. Knoop and A. Prantl, editors, *Post-Proceedings of 15. Kolloquium Programmiersprachen und Grundlagen der Programmierung*, Technische Universität Wien, 2009, to appear.

[18] von Bertalanffy, L., "General Systems Theory: Foundations, Development, Applications," 1968.

From Coalgebraic to Monoidal Traces

Bart Jacobs

Institute for Computing and Information Sciences,
Radboud University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
Email: bart@cs.ru.nl *URL:* www.cs.ru.nl/~bart

Abstract

The main result of this paper shows how coalgebraic traces, in suitable Kleisli categories, give rise to traced monoidal structure in those Kleisli categories, with finite coproducts as monoidal structure. At the heart of the matter lie partially additive monads inducing partially additive structure in their Kleisli categories. By applying the standard "Int" construction one obtains compact closed categories for "bidirectional monadic computation".

Keywords: Coalgebra, execution trace, monoidal trace, iteration, Kleisli category

1 Introduction

The notion of trace occurs prominently in the (classical) categorical work on traced monoidal categories [13]. It generalises the trace operator in linear algebra and captures fixed points for operations with feedback. Recently, also a coalgebraic approach to traces emerged [12], where traces are maps in Kleisli categories induced by monads that capture the observable behaviour in for instance sequences of (monadic) computations. Such traces are often described by removing states from execution traces. Naturally one wonders if there is a connection between these monoidal and coalgebraic traces. This paper addresses this question and shows how coalgebraic traces give rise to monoidal traces. The word 'trace' thus different meanings in this context, but hopefully without generating too much confusion.

The way this result is obtained is via the work of Haghverdi [9], where it is shown that partially additive categories (see also [5]) are traced monoidal,

⋆ ENTCS Proceedings of *Coalgebraic Methods in Computer Science* (CMCS 2010).

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

via what is called the execution (or trace) formula. Thus the paper proceeds by proving that under certain assumptions on a monad T, firstly the Kleisli category of T is such a partially additive category, and secondly the execution formula coincides with the coalgebraic trace. The technical core of the paper involves the identification of the notion of a "partially additive monad", see Definition 4.3, and the proof that the Kleisli categories of such monads are partially additive.

We describe the organisation of this paper and at the same time the flow of developments. The paper starts with an elementary initial algebra in Section 2 that gives rise to a final coalgebra in suitably order-enriched Kleisli categories in Section 3, and thus to coalgebraic trace semantics, following [12]. For this particular coalgebra it also yields an iteration operation as in [8,6]. Section 4 then shows that what we call partially additive monads in such a setting additionally yields partially additive structure $\amalg$ on Kleisli homsets, as studied earlier in [5]. They enable us to obtain the main result in Section 5, namely that Kleisli categories of suitable monads, with finite coproducts, are traced monoidal, via [9]. The "Int" construction from [13] can then be applied and yields in Section 6 new categories $\mathcal{B}d(T)$ of "bidirectional monadic computations", with connections to game semantics and quantum computation. This forms a topic of its own that will be further investigated elsewhere. Throughout the paper there is a series of running examples, consisting of powerset, lift, distribution and quantale monads. The latter eventually yields examples of strongly compact closed categories.

2 A basic initial algebra

Assume $\mathbb{C}$ is a category with countable coproducts, written as $\coprod_{i\in I} X_i$ with coprojections $\kappa_i \colon X_i \to \coprod_{i\in I} X_i$. In order to further fix the notation, we shall write $[]_X \colon 0 \to X$ or simply $[] \colon 0 \to X$ (without subscript) for the unique arrow (the empty cotuple) out of an initial object 0. The two coprojections for a binary coproduct are written as $X \xrightarrow{\kappa_\ell} X + Y \xleftarrow{\kappa_r} Y$, with cotupling of $f \colon X \to Z$ and $g \colon Y \to Z$ denoted by $[f, g] \colon X + Y \to Z$. Hence on morphisms, $h + k = [\kappa_\ell \circ h, \kappa_r \circ k]$.

This $\mathbb{C}$ with its finite coproducts $(0, +)$ yields a symmetric monoidal category (SMC). In general, for an SMC $(\mathbb{A}, I, \otimes)$ we write the familiar isomorphisms as:

$$(1) \qquad X \otimes (Y \otimes Z) \xrightarrow[\cong]{\alpha} (X \otimes Y) \otimes Z \quad X \otimes I \xrightarrow[\cong]{\rho} X \quad X \otimes Y \xrightarrow[\cong]{\gamma} Y \otimes X$$

A copower $I \cdot X = \coprod_{i\in I} X$ comes with coprojections $\kappa_i \colon X \to I \cdot X$ and cotupling $[f_i]_{i\in I} \colon I \cdot X \to Y$ for an I-indexed collection of maps $f_i \colon X \to Y$.

Proposition 2.1 *Let $\mathbb{C}$ have countable coproducts, as above. For a fixed ob-*

ject $Y \in \mathbb{C}$, the functor $Y + (-)\colon \mathbb{C} \to \mathbb{C}$ has the copower $\mathbb{N} \cdot Y = \coprod_{n\in\mathbb{N}} Y$ as initial algebra, with structure map:

$$Y + \mathbb{N}\cdot Y \xrightarrow[\cong]{\xi \overset{defn}{=} \left[\kappa_0, [\kappa_{n+1}]_{n\in\mathbb{N}}\right]} \mathbb{N}\cdot Y$$

Proof For an arbitrary algebra $[a, b]\colon Y + X \to X$ we define $f\colon \mathbb{N}\cdot Y \to X$ as $f = [b^n \circ a]_{n\in\mathbb{N}}$. It forms the unique algebra homomorphism from ξ to $[a, b]$. □

The copower object $\mathbb{N}\cdot Y$ may be understood in the standard way (see [16]) as the colimit of repeated application of the functor $Y+(-)$ to the initial object $0 \in \mathbb{C}$, as in:

$$0 \xrightarrow{[]} 1\cdot Y \xrightarrow{Y+[]} 2\cdot Y \xrightarrow{Y+(Y+[])} 3\cdot Y \longrightarrow \cdots$$

We write $0 \cdot Y = 0$ and $(n + 1) \cdot Y = Y + n \cdot Y$. The resulting colimit cone $\lambda_n\colon n \cdot Y \to \mathbb{N} \cdot Y$ is then defined as:

$$\lambda_0 = []\colon 0 \longrightarrow \mathbb{N}\cdot Y \quad \text{and} \quad \lambda_{n+1} = [\kappa_n, \lambda_n]\colon Y + n\cdot Y \longrightarrow \mathbb{N}\cdot Y. \tag{2}$$

The "twist" in this definition of λ_n is needed to ensure that the "oldest" element in $n \cdot Y$ is put at the first position in $\mathbb{N} \cdot Y$. Indeed, in this way we get $\lambda_{n+1} \circ \kappa_r = \lambda_n$ for the chain maps $\kappa_r\colon Y_n \to Y_{n+1}$.

3 A final coalgebra in a Kleisli category: trace semantics

We now assume that our category $\mathbb{C}$ (with coproducts) carries a monad $T\colon \mathbb{C} \to \mathbb{C}$, with unit η and multiplication μ. We shall write $\mathcal{K}\ell(T)$ for the resulting Kleisli category, with forgetful functor $\mathcal{K}\ell(T) \to \mathbb{C}$ and left adjoint $J\colon \mathbb{C} \to \mathcal{K}\ell(T)$. Trivially, $\mathcal{K}\ell(T)$ inherits coproducts from $\mathbb{C}$. They behave like in $\mathbb{C}$ on objects, but have slightly different coprojections and coproducts of maps. In order to disambiguate them we shall write a dot for operations in a Kleisli category, as in:

$$\begin{aligned} g \odot f &= \mu \circ Tg \circ f \\ \dot{\kappa}_\ell = J(\kappa_\ell) &= \eta \circ \kappa_\ell \\ h \dot{+} k &= [T(\kappa_\ell) \circ h, T(\kappa_r) \circ k], \quad \text{so that} \quad J(a + b) = J(a) \dot{+} J(b). \end{aligned}$$

This dot-notation is meant to prevent confusion. We shall use it with prudence and shall write for instance identity maps in Kleisli categories simply as id_X

and not as $\dot{\mathrm{id}}_X = \eta_X$. The (obvious) identities $g \odot J(f) = g \circ f$ and $J(g) \odot f = T(g) \circ f$ are often used.

For an object $Y \in \mathbb{C}$ we thus also get a functor $Y + (-) \colon \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$. Its initial algebra is the copower $\mathbb{N} \cdot Y$, by Proposition 2.1, but in $\mathcal{K}\ell(T)$. Its final coalgebra will be of more interest here.

In [12] a general framework is developed for generic trace semantics, which works for coalgebras of the form $X \to TFX$, where T is a monad and F an endofunctor. The main result in [12] says that, under suitable order-theoretic assumptions, the initial algebra in $\mathbb{C}$ yields a final coalgebra in $\mathcal{K}\ell(T)$. Here we shall only be interested in the special case where the functor F is of the form $Y + (-)$.

Proposition 3.1 (From [12]) *Let T be a monad on a category $\mathbb{C}$ with coproducts. Assume that the Kleisli category $\mathcal{K}\ell(T)$ is dcpo-enriched, that (Kleisli) homsets have bottom elements $\bot$ which are left strict (i.e. satisfy $\bot \odot f = \bot$) and that cotupling is monotone (i.e. $[-,-]$ preserves the order in both coordinates).*

The initial algebra $\xi \colon Y + \mathbb{N} \cdot Y \xrightarrow{\cong} \mathbb{N} \cdot Y$ in $\mathbb{C}$ from Proposition 2.1 then yields a final coalgebra $J(\xi^{-1}) \colon \mathbb{N} \cdot Y \xrightarrow{\cong} T(Y + \mathbb{N} \cdot Y)$ of the functor $Y + (-) \colon \mathcal{K}\ell(T) \to \mathcal{K}\ell(T)$. Concretely, this means that for every coalgebra $c \colon X \to T(Y + X)$ there is a unique map $\mathrm{tr}(c) \colon X \to T(\mathbb{N} \cdot Y)$ forming a unique coalgebra homomorphism in the Kleisli category $\mathcal{K}\ell(T)$ as in:

$$\begin{array}{ccc} Y + X & \overset{id \dot{+} \mathrm{tr}(c)}{\dashrightarrow} & Y + \mathbb{N} \cdot Y \\ {\scriptstyle c}\uparrow & & \cong\uparrow{\scriptstyle J(\xi^{-1})} \\ X & \overset{\mathrm{tr}(c)}{\dashrightarrow} & \mathbb{N} \cdot Y \end{array} \tag{3}$$

Intuitively, this trace map $\mathrm{tr}(c)$ sends an element $x \in X$ to the "set" of those $(n, y) \in \mathbb{N} \cdot Y$ for which c reaches $y \in Y$ from x in n cycles through X, see the examples below.

We shall write $c^\# = \nabla \odot \mathrm{tr}(c) \colon X \to Y$ in $\mathcal{K}\ell(T)$ for the "iterate" of c, like in [8,5] [1], where $\nabla = [\mathrm{id}]_{n \in \mathbb{N}} \colon \mathbb{N} \cdot Y \to Y$ is the codiagonal in $\mathcal{K}\ell(T)$. It yields an operator between Kleisli homsets of the form:

$$\mathcal{K}\ell(T)(X, Y + X) \xrightarrow{(-)^\#} \mathcal{K}\ell(T)(X, Y)$$

Clearly, such an iterate $c^\#$ does not keep track of the number of rounds that are made to reach a result in Y—like $\mathrm{tr}(c)$ does.

Here we omit the proof and refer to [12] for details but we shall explicitly describe the definition of the trace map $\mathrm{tr}(c)$ so that we can use it later on. It

[1] In [8,5] the notation $c^\dagger$ is used, instead of $c^\#$, but we prefer to reserve the dagger † for involutions, see Lemma 5.4.

uses the fact that the initial object $0 \in \mathbb{C}$ is final in Kleisli categories as in the proposition, with $\bot\colon X \to 0$ in $\mathcal{K}\ell(T)$ as unique map (see also Lemma 4.1 (1) below). This allows us to define a sequence of maps $c_n\colon X \to n \cdot Y$ in $\mathcal{K}\ell(T)$ as:

$$\begin{cases} c_0 = \bot : X \longrightarrow 0 = 0 \cdot Y \\ c_{n+1} = (\mathrm{id} \mathbin{\dot{+}} c_n) \circ c : X \longrightarrow Y + X \longrightarrow Y + n \cdot Y = (n+1) \cdot Y \end{cases} \tag{4}$$

Then we can define the trace map as join:

$$\mathrm{tr}(c) = \bigvee\nolimits_{n \in \mathbb{N}} J(\lambda_n) \circ c_n \tag{5}$$

in the Kleisli homset of maps $X \to \mathbb{N} \cdot Y$, with λ_n as defined in (2).

Example 3.2 We shall consider what the above result amounts to for our four main examples for the monad T, namely $\mathcal{P}, \mathcal{D}, \mathcal{L}$ and $Q^{(-)}$ on **Sets**.

(1) The Kleisli category $\mathcal{K}\ell(\mathcal{P})$ of the powerset monad $\mathcal{P}\colon \mathbf{Sets} \to \mathbf{Sets}$ is the category of sets with relations as arrows between them. Homsets are ordered by pointwise inclusion, and form complete lattices. Commutation of diagram (3) means that for a coalgebra $c\colon X \to \mathcal{P}(Y + X)$ the resulting trace map $\mathrm{tr}(c)\colon X \to \mathcal{P}(\mathbb{N} \cdot Y)$ satisfies:

$$\begin{aligned} (n, y) \in \mathrm{tr}(c)(x_0) &\Leftrightarrow \exists x_1, \ldots, x_n \in X.\, x_1 \in c(x_0) \wedge \ldots \wedge x_{n-1} \in c(x_n) \wedge y \in c(x_n) \\ &\Leftrightarrow \exists x_1, \ldots, x_n \in X.\, \textstyle\bigwedge_{i<n} x_{i+1} \in c(x_i) \wedge y \in c(x_n) \end{aligned}$$

where we have left out the coprojections κ_ℓ, κ_r for simplicity.

(2) For the lift monad $\mathcal{L} = 1 + (-)$ we write $\bot \in 1 + X$ for the bottom element $\bot \in 1$ and $\mathrm{up}(x) \in 1 + X$ for an element $x \in X$. These sets $1 + X$ are "flat" dcpos. For $c\colon X \to 1 + (Y + X)$ we then get a trace map $\mathrm{tr}(c)\colon X \to 1 + \mathbb{N} \cdot Y$ with:

$$\mathrm{tr}(c)(x_0) = \mathrm{up}(n, y) \Leftrightarrow \exists x_1, \ldots, x_n \in X.\, \textstyle\bigwedge_{i<n} c(x_i) = \mathrm{up}(x_{i+1}) \wedge c(x_n) = \mathrm{up}(y)$$

(3) We shall write $\mathcal{D}$ for the (sub)distribution monad on **Sets** given by:

$$\mathcal{D}(X) = \{\varphi\colon X \to [0,1] \mid \textstyle\sum_{x \in X} \varphi(x) \leq 1\}.$$

Notice that we do not require that such $\varphi \in \mathcal{D}(X)$ have finite support (*i.e.* have finitely many elements $x \in X$ that are not mapped to 0). The sets $\mathcal{D}(X)$ are dcpos with pointwise order and bottom element $\bot = \lambda x.\, 0$. The Kleisli maps $X \to \mathcal{D}(Y)$ can then also be ordered, pointwise.

For a coalgebra $c\colon X \to \mathcal{D}(Y + X)$ we obtain a trace map $\mathrm{tr}(c)\colon X \to \mathcal{D}(\mathbb{N} \cdot$

Y) as in diagram (3), given explicitly by the following probability formula.

$$\begin{aligned}\mathrm{tr}(c)(x_0)(n,y) &= \sum_{x_1,\ldots,x_n\in X} c(x_0)(x_1)\cdot\ldots\cdot c(x_{n-1})(x_n)\cdot c(x_n)(y)\\ &= \sum_{x_1,\ldots,x_n\in X}\prod_{i<n} c(x_i)(x_{i+1})\cdot c(x_n)(y)\end{aligned}$$

(4) Let Q be a quantale, *i.e.* a complete lattice with a monoid structure $(1,\cdot)$ where multiplication $\cdot$ preserves suprema $\bigvee$ in both arguments (see [14]). The mapping $X \mapsto Q^X$ is then a monad on **Sets** with unit and multiplication given by:

$$\begin{array}{ccc} X \xrightarrow{\quad\eta\quad} Q^X & & Q^{(Q^X)} \xrightarrow{\quad\mu\quad} Q^X \\ x \longmapsto \lambda x'.\begin{cases} 1 & \text{if } x'=x \\ \bot & \text{otherwise}\end{cases} & & \Phi \longmapsto \lambda x.\ \bigvee_{\varphi\in Q^X} \Phi(\varphi)\cdot\varphi(x)\end{array}$$

A function $f\colon X\to Y$ yields $Q^f\colon Q^X\to Q^Y$ by $\varphi\mapsto \lambda y.\ \bigvee_{x\in f^{-1}(y)}\varphi(x)$. The powerset monad $\mathcal{P}$ from (1) is a special case for $Q=2$.

For a coalgebra $c\colon X\to Q^{Y+X}$ diagram (3) now yields a trace map $\mathrm{tr}(c)\colon X\to Q^{\mathbb{N}\cdot Y}$ that formally resembles the previous one:

$$\mathrm{tr}(c)(x_0)(n,x_{n+1}) = \bigvee_{x_1,\ldots,x_n\in X}\prod_{i\leq n} c(x_i)(x_{i+1})$$

We collect some basic results about coalgebraic traces $\mathrm{tr}(c)$ and iterates $c^\#$.

Lemma 3.3 *In the situation of the previous proposition:*

(i) *Uniformity: if f is a homomorphism of coalgebras $c\to d$ in $\mathcal{K}\ell(T)$,*

$$\mathrm{tr}(c) = \mathrm{tr}(d)\circ f \quad \textit{and so} \quad c^\# = d^\#\circ f.$$

(ii) *Naturality in Y: for $g\colon Y\to T(V)$,*

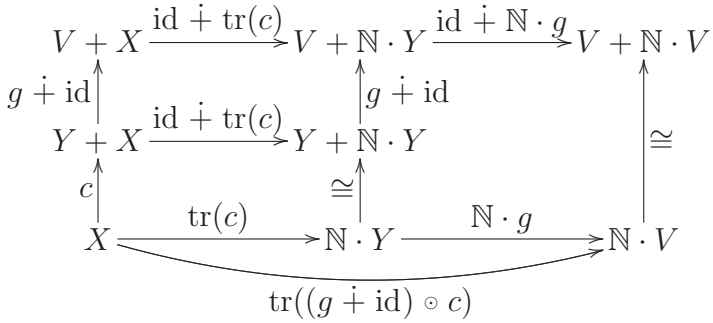
$$\mathrm{tr}((g \mathbin{\dot{+}} id)\odot c) = \mathbb{N}\cdot g\odot \mathrm{tr}(c) \quad \textit{and} \quad ((g\mathbin{\dot{+}} id)\odot c)^\# = g\odot c^\#.$$

(iii) *Dinaturality in X: for $f\colon U\to T(X)$,*

$$\mathrm{tr}(c\odot f) = \mathrm{tr}((id\mathbin{\dot{+}} f)\odot c)\circ f \quad \textit{and} \quad (c\odot f)^\# = ((id\mathbin{\dot{+}} f)\odot c)^\#\odot f.$$

Proof Everything follows from (the uniqueness part of) finality. For instance

the second point involves the diagram:



The diagram on the right commutes by definition of $\mathbb{N} \cdot g$. □

4 Additive structure on Kleisli homsets

We start this section by some preparatory observations about the structure induced by order on Kleisli homsets, making coproducts behave a bit like products (*i.e.* biproducts). It will lead to a description of additive structure (certain sums) in such homsets, which we shall write with a separate symbol $\amalg$ in order to prevent confusion with the sum $f + g = [\kappa_\ell \circ f, \kappa_r \circ g]$ induced by coproducts $+$. The main contribution of this section lies in the notion of partially additive monad, see Definition 4.2, and in the result that the Kleisli categories of such monads form partially additive categories.

The first point of the next lemma has already been used, but will be repeated here for completeness.

Lemma 4.1 *Assume $\mathbb{C}$ is a category with countable coproducts. Let $T \colon \mathbb{C} \to \mathbb{C}$ be a monad whose Kleisli homsets $\mathcal{K}\ell(T)(X, Y) = \mathbb{C}(X, T(Y))$ are partially ordered.*

(i) *If each Kleisli homset has a bottom element $\bot \colon X \to T(Y)$ which is left strict (i.e. satisfies $\bot \circ f = \bot$), then 0 is a final object in $\mathcal{K}\ell(T)$. Since 0 is obviously initial in $\mathcal{K}\ell(T)$, it becomes a zero object (or "nullary" biproduct).*

(ii) *If $\bot$ is "bi-strict", i.e. is preserved by both pre- and post-composition in $\mathcal{K}\ell(T)$, then there are natural "projection" maps $p_j \colon \coprod_{i \in I} X_i \to T(X_j)$ satisfying:*

$$p_j \circ \dot{\kappa}_j = id \quad \text{and} \quad p_j \circ \dot{\kappa}_m = \bot \quad \text{for } j \neq m.$$

In the binary case we shall write p_ℓ, p_r, just like for coprojections κ_ℓ, κ_r.

Proof (1) There is only $\bot \colon X \to 0$ in $\mathcal{K}\ell(T)$ because each $f \colon X \to 0$ satisfies: $f = f \circ \mathrm{id}_0 = f \circ \bot = \bot$, by left strictness.

(2) One takes $p_j = [p_{i,j}]_{i\in I} \colon \coprod_{i\in I} X_i \to T(X_j)$ where $p_{j,j} = \eta_{X_j}$ and $p_{i,j} = \bot$ for $i \neq j$. Then clearly $p_j \odot \dot{\kappa}_j = p_j \circ \kappa_j = p_{j,j} = \eta$, which is the identity in $\mathcal{K}\ell(T)$, and $p_j \odot \dot{\kappa}_m = \bot$ for $j \neq m$. Naturality follows from (right) strictness. □

For the formulation of the following notion it is convenient to assume that our category $\mathbb{C}$ has set-indexed products. The definition can be given without such products, using "jointly monic families". But that only makes it harder to understand the matter.

Definition 4.2 Assume projections p_i as in the previous lemma, for a monad T on a category $\mathbb{C}$ with countable coproducts and products. By bc, for 'bi-cartesian', we denote the following map.

$$\text{(6)} \qquad \text{bc} = \Big(T(\textstyle\coprod_{i\in I} X_i) \xrightarrow{\langle\, p_i^\flat \,\rangle_{i\in I}} \prod_{i\in I} T(X_i)\Big) \quad \text{where} \quad p_i^\flat = \mu \circ T(p_i).$$

The monad T is called partially additive if these bc's form cartesian natural transformations with monic components. This means that all naturality squares:

$$\begin{array}{ccc} T(\coprod_i X_i) & \xrightarrow{T(\coprod_i f_i)} & T(\coprod_i Y_i) \\ \downarrow{\scriptstyle \text{bc}} & & \downarrow{\scriptstyle \text{bc}} \\ \prod_i T(X_i) & \xrightarrow{\prod_i T(f_i)} & \prod_i T(Y_i) \end{array}$$

are pullbacks in $\mathbb{C}$, for collections of maps $f_i \colon X_i \to Y_i$ in $\mathbb{C}$.

The monad T may be called additive if these bc's are isomorphisms. Such monads are investigated further in [7]. The next definition of sums on Kleisli homsets is based on [5].

Definition 4.3 Let T be a partially additive monad on $\mathbb{C}$, as in the previous definition. For countably many $f_i \colon X \to Y$ in $\mathcal{K}\ell(T)$ write $\amalg_{i\in I} f_i = \nabla_I \odot b \colon X \to Y$ in $\mathcal{K}\ell(T)$ if there is a "bound" map $b \colon X \to T(I \cdot Y) = T(\coprod_{i\in I} Y)$ with $p_i \odot b = f_i$.

This bound property can be expressed as: $\text{bc} \circ b = \langle f_i \rangle_{i\in I} \colon X \to \prod_{i\in I} T(Y) = T(Y)^I$. By the mono requirement on bc there is at most one such bound b.

We may observe that certain joins always exist: for a map $f \colon X \to T(Y + Z)$, one has $f = (\dot{\kappa}_\ell \odot p_\ell \odot f) \amalg (\dot{\kappa}_r \odot p_r \odot f)$, via the bound $(\dot{\kappa}_\ell \mathbin{\dot{+}} \dot{\kappa}_r) \odot f \colon X \to T((Y + Z) + (Y + Z))$.

Before further investigation of this sum $\amalg$ we check what it means in the examples.

Example 4.4 We shall consider the powerset monad as special case of the quantale monad $Q^{(-)}$, namely for $Q = 2$. For convenience, we consider the binary sum $\amalg$ only.

(1) For the lift monad $\mathcal{L}$, recall that Kleisli homsets are flat orders, in which very few joins (or sums) exist. The projections $Y_\ell + Y_r \to 1 + Y_i$ are given by $p_i(w) = \mathrm{up}(y)$ iff $w = \kappa_i(y)$, for $i \in \{\ell, r\}$. For $b\colon X \to 1 + (Y + Y)$ one has:

$$(p_i \odot b)(x) = \begin{cases} \mathrm{up}(y) & \text{if } b(x) = \mathrm{up}(\kappa_i y) \\ \bot & \text{otherwise.} \end{cases}$$

Hence b is completely determined by these $p_i \odot b$, so that projections are jointly monic—and bc from (6) is monic. The pullback property for bc is left to the reader.

Now if $f_i\colon X \to 1 + Y$ are given, and we have a bound $b\colon X \to 1 + (Y + Y)$ with $p_i \odot b = f_i$, then we know:

- if $f_\ell(x) = \mathrm{up}(y)$, then $(p_\ell \odot b)(x) = \mathrm{up}(y)$ so that $b(x) = \mathrm{up}(\kappa_\ell y)$ and thus $(p_r \odot b)(x) = \bot$, so that $f_r(x) = \bot$.
- if $f_r(x) = \mathrm{up}(y)$, then similarly $f_\ell(x) = \bot$.

The existence of this bound b thus guarantees that both $f_\ell(x) \neq \bot$ and $f_r(x) \neq \bot$ does not happen. Hence their join exists, namely the non-bottom value, if any. This value is given by $\nabla \odot b$.

(2) The Kleisli category $\mathcal{K}\ell(\mathcal{D})$ of the subdistribution monad $\mathcal{D}$ inherits its pointwise order from the unit interval $[0, 1]$. This interval has joins, but it turns out that $\amalg$ describes the partially defined $+$ on $[0, 1]$. The projections $Y_\ell + Y_r \to \mathcal{D}(Y_i)$ are given by $p_i(w)(y) = \mathsf{if}\ w = \kappa_i y\ \mathsf{then}\ 1\ \mathsf{else}\ 0$. Thus for $b\colon X \to \mathcal{D}(Y + Y)$ we have $(p_i \odot b)(x)(y) = \sum_{w \in Y+Y} p_i(w)(y) \cdot b(x)(w) = b(x)(\kappa_i y)$. And $\mathrm{bc}\colon \mathcal{D}(Y_\ell + Y_r) \to \mathcal{D}(Y_\ell) \times \mathcal{D}(Y_r)$ is given by $\mathrm{bc}(\varphi) = \langle \varphi \circ \kappa_\ell, \varphi \circ \kappa_r \rangle$. It is thus clearly monic.

For the pullback property for bc, assume a collection $f_i\colon X_i \to Y_i$ together with maps $\langle \alpha_\ell, \alpha_r \rangle\colon A \to \mathcal{D}(X_\ell) \times \mathcal{D}(X_r)$ and $\beta\colon A \to \mathcal{D}(Y_\ell + Y_r)$ satisfying $\mathcal{D}(f_i) \circ \alpha_i = p_i^\flat \circ \beta$. The only possible mediating map $\gamma\colon A \to \mathcal{D}(X_\ell + X_r)$ is defined as $\gamma(a)(\kappa_\ell x) = \alpha_\ell(a)(x)$ and $\gamma(a)(\kappa_r x) = \alpha_r(a)(x)$. We have to check that $\gamma(a)$ is a subdistribution. This follows from because $\beta(a)$ is a subdistribution:

$$\begin{aligned} 1 \geq \textstyle\sum_z \beta(a)(z) &= \textstyle\sum_{y \in Y_\ell} \beta(a)(\kappa_\ell y) + \sum_{y \in Y_r} \beta(a)(\kappa_r y) \\ &= \textstyle\sum_{y \in Y_\ell} (p_\ell^\flat \circ \beta)(a)(y) + \sum_{y \in Y_r} (p_r^\flat \circ \beta)(a)(y) \\ &= \textstyle\sum_{y \in Y_\ell} (\mathcal{D}(f_\ell) \circ \alpha_\ell)(a)(y) + \sum_{y \in Y_r} (\mathcal{D}(f_r) \circ \alpha_r)(a)(y) \\ &= \textstyle\sum_{y \in Y_\ell} \sum_{x \in f_\ell^{-1}(y)} \alpha_\ell(a)(x) + \sum_{y \in Y_r} \sum_{x \in f_r^{-1}(y)} \alpha_r(a)(x) \\ &= \textstyle\sum_{x \in X_\ell} \alpha_\ell(a)(x) + \sum_{x \in X_r} \alpha_r(a)(x) \\ &= \textstyle\sum_{w \in X_\ell + X_r} \gamma(a)(w). \end{aligned}$$

Further, if $f_i \colon X \to \mathcal{D}(Y)$ are given with $f_i = p_i \odot b$, then:

$$\begin{aligned}(f_\ell \amalg f_r)(x)(y) \;=\; (\nabla \odot b)(x)(y) &= \textstyle\sum_{w \in Y+Y} \nabla(w)(y) \cdot b(x)(w) \\ &= b(x)(\kappa_\ell y) + b(x)(\kappa_r y) \\ &= f_\ell(x)(y) + f_r(x)(y).\end{aligned}$$

(3) For the quantale monad $Q^{(-)}$ we have projections $Y_\ell + Y_r \to Q^{Y_i}$ given by $p_i(w)(y) = \mathsf{if}\ w = \kappa_i y\ \mathsf{then}\ 1\ \mathsf{else}\ \bot$, so that for $b \colon X \to Q^{Y+Y}$ we get $(p_i \odot b)(x)(y) = \bigvee_{w \in Y+Y} p_i(w)(y) \cdot b(x)(w) = b(x)(\kappa_i y)$. The map bc is in this case an isomorphism $Q^{Y_\ell + Y_r} \xrightarrow{\cong} Q^{Y_\ell} \times Q^{Y_r}$, so that $Q^{(-)}$ is an additive monad. And if the f_i have a bound, then their sum is given by union: $(f_\ell \amalg f_r)(x)(y) = f_\ell(x)(y) \vee f_r(x)(y)$.

These examples illustrate that the sum operation $\amalg$ is determined by (Kleisli) composition, and hence ultimately by the monad involved.

We continue with some basic properties of $\amalg$.

Lemma 4.5 *In the situation of the previous definition, one has:*

(i) $\amalg$ *is preserved by both pre- and post-composition;*

(ii) *The sum of the singleton family* $\{f\}$ *if* f *itself; the sum over the empty family is* $\bot$*;*

(iii) *If cotupling* $[-,-]$ *is monotone, then* $f_j \leq \amalg_{i \in I} f_i$*;*

(iv) *Assume the Kleisli category is* ***Dcpo****-enriched. Let* I *be a countable set such that* $\amalg_{i \in J} f_i$ *exists for each finite subset* $J \subseteq I$*. Then* $\amalg_{i \in I} f_i$ *exists.*

Proof (1) Suppose $\amalg_i f_i$ exists for $f_i \colon X \to T(Y)$, say with bound $b \colon X \to T(I \cdot Y)$. For $g \colon U \to T(X)$ the composite $b \odot g \colon U \to T(I \cdot Y)$ is obviously a bound for $f_i \odot g$ and yields $\amalg_i(f_i \odot g) = \nabla \odot b \odot g = (\amalg_i f_i) \odot g$.

Similarly, for $h \colon Y \to T(U)$ the map $I \cdot h \odot b$ is a bound for $h \odot f_i$, by naturality of projections, so that $\amalg_i(h \odot f_i) = \nabla \odot I \cdot h \odot b = h \odot \nabla \odot b = h \odot (\amalg_i f_i)$.

(2) The map f is a bound for $\{f\}$ and $\bot$ is a bound for the empty family.

(3) If cotupling is monotone we get $p_i \leq \nabla$ and thus for a bound b,

$$f_i = p_i \odot b \leq \nabla \odot b = \amalg_i f_i.$$

(4) Assume for convenience that our index set is $\mathbb{N}$. Let $f_n \colon X \to T(Y)$, for $n \in \mathbb{N}$, be a collection such that the sum $\amalg$ exists for each finite subset. There are sums $f_0 \amalg f_1 \amalg \cdots \amalg f_{n-1}$, say via bound $b_n \colon X \to T(n \cdot Y)$. It is not hard to see that the collection $\dot{\kappa}_i \odot f_i \colon X \to T(\mathbb{N} \cdot Y)$, for $i < n$, also has a

bound, namely $b'_n = (\dot\kappa_0 \mathbin{\dot+} \cdots \mathbin{\dot+} \dot\kappa_{n-1}) \circ b_n \colon X \to T(n \cdot \mathbb{N} \cdot Y)$. We then define

$$g_n = \nabla \circ b'_n = (\dot\kappa_0 \circ f_0) \amalg \cdots \amalg (\dot\kappa_{n-1} \circ f_{n-1}) : X \longrightarrow \mathbb{N} \cdot Y.$$

This yields a monotone collection $g_n \leq g_{n+1}$ by the previous point. Hence we get a map $f = \bigvee_n g_n \colon X \to \mathbb{N} \cdot Y$ as directed join, which is the intended sum. □

One further property of $\amalg$ is required, which is sometimes called "partition associativity". It is non-trivial and depends on the pullback requirement from Definition 4.2.

Lemma 4.6 *If a (countable) collection I can be written as disjoint union $I = \bigcup_{k\in K} I_k$, then $\amalg_{i\in I} f_i$ exists if and only each sum $f_k = \amalg_{i\in I_k} f_i$ exists and $\amalg_{i\in I} f_i = \amalg_{k\in K} f_k$.*

As a result, $\amalg$ is commutative and associative.

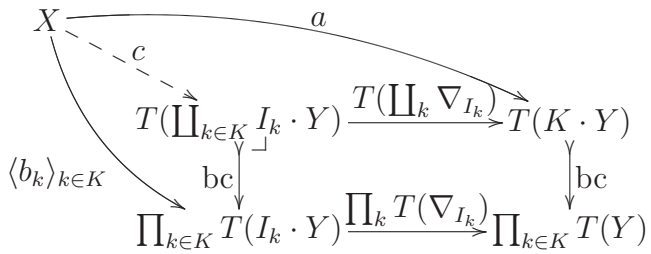
Proof If $I = \bigcup_{k\in K} I_k$ is a disjoint union, then $I \cdot Y \cong \coprod_{k\in K} I_k \cdot Y$. Hence it is more convenient to consider a collection of maps $f_{k,i} \colon X \to Y$ for $k \in K$ and $i \in I_k$.

In one direction, suppose $b \colon X \to \coprod_{k\in K} I_k \cdot Y$ is bound for the collection $(f_{k,i})$, so that $f_{k,i} = p_i \circ p_k \circ b$. Write $b_k = p_k \circ b \colon X \to I_k \cdot Y$. It forms a bound for the collection $(f_{k,i})_{i\in I_k}$, since $p_i \circ b_k = p_i \circ p_k \circ b = f_i$, for each $i \in I_k$. The sums $f_k = \amalg_{i\in I_k} f_i = \nabla_{I_k} \circ b_k$ have a bound $a = (\coprod_{k\in K} \nabla_{I_k}) \circ b \colon X \to K \cdot Y$, since for each $k \in K$,

$$\begin{aligned} p_k \circ a \;=\; p_k \circ (\textstyle\coprod_{k\in K} \nabla_{I_k}) \circ b &= \nabla_{I_k} \circ p_k \circ b \qquad \text{by naturality of projections} \\ &= \nabla_{I_k} \circ b_k \;=\; \amalg_{i\in I_k} f_i \;=\; f_k. \end{aligned}$$

Hence $\amalg_{k\in K} f_k$ exists as $\nabla_K \circ a = \nabla_K \circ (\coprod_{k\in K} \nabla_{I_k}) \circ b = \nabla_I \circ b = \amalg_{i\in I} f_i$.

For the other direction assume that the sums $f_k = \amalg_{i\in I_k} f_{k,i}$ and $\amalg_{k\in K} f_k$ exist; we need to show that also $\amalg_{k\in I, i\in I} f_{k,i}$ exists—and is equal to $\amalg_{k\in K} f_k$. So let $b_k \colon X \to I_k \cdot Y$ be a bound for the collection $(f_{k,i})_{i\in I_k}$ and $a \colon X \to K \cdot Y$ be a bound for these $f_k = \coprod_{i\in I_k} f_i = \nabla_{I_k} \circ b_k$. We need a bound $c \colon X \to \coprod_{k\in K} I_k \cdot Y$, which we obtain via the following naturality pullback, as required in Definition 4.3.



Hence the mediating map c is a bound for these b_k and thus for the $f_{k,i}$. The resulting sum is: $\amalg_{k\in K, i\in I_k} f_{k,i} = \nabla_K \circ \coprod_{k\in K} \nabla_{I_k} \circ c = \nabla_K \circ a = \amalg_{k\in K} f_k$. □

We are now ready to collect the requirements that we need in this paper.

Requirements 4.7 *The category $\mathbb{C}$ is assumed to have countable coproducts and the monad $T\colon \mathbb{C} \to \mathbb{C}$ satisfies:*

(i) *its Kleisli category $\mathcal{K}\ell(T)$ is $\boldsymbol{Dcpo}_\perp$-enriched, so that Kleisli homsets have (countable) directed joins and a bottom element, which are preserved by composition;*

(ii) *this Kleisli category also has monotone cotupling;*

(iii) *the monad T is partially additive, as in Definition 4.3.*

¿From Lemma 4.5 we may now conclude a basic result.

Proposition 4.8 *Let category $\mathbb{C}$ with monad T satisfy Requirement 4.7. The Kleisli category $\mathcal{K}\ell(T)$ is then partially additive. Further, it is additive (has all countable sums $\amalg$) iff it has countable strict biproducts.* □

For what it precisely means to be partially additive we refer to the literature [5]. Here we shall simply use that Kleisli homsets have certain sums $\amalg$, with properties as described in Lemma 4.5. The projections p_i make the Kleisli categories into what are called 'unique decomposition categories', see also [10]. The "further" part of the proposition is [9, Theorem 3.0.17]. It applies to the Kleisli category of quantale monads.

5 Kleisli categories are traced monoidal

Now that we have seen additive structure on Kleisli homsets we can conclude from [9] that we have traced monoidal structure in these Kleisli categories. But before we do so we return to Section 3 and re-describe the iterate $c^\#$ of a coalgebra c in terms of the newly discovered sums. This will be used (in the proof of Theorem 5.2) to show that the induced traced monoidal structure coincides with the coalgebraic trace.

Lemma 5.1 *For $\mathbb{C}, T$ satisfying Requirements 4.7 the iterate $c^\#$ of a coalgebra $c\colon X \to T(Y + X)$, from Proposition 3.1, can be described as sum:*

$$c^\# = c_\ell \circ \amalg_{n\in\mathbb{N}} c_r^n = c_\ell \circ c_r^\star,$$

where $c_\ell = p_\ell \circ c\colon X \to T(Y)$ and $c_r = p_r \circ c\colon X \to T(X)$, and $h^\star = \amalg_{n\in\mathbb{N}} h^n$.

Proof Recall that the iterate is defined as $c^\# = \nabla \circ \mathrm{tr}(c)\colon X \to \mathbb{N}\cdot Y \to Y$. Hence it is a sum $\amalg$ by construction. So we only have to check that $p_i \circ \mathrm{tr}(c) = c_\ell \circ c_r^i$, for $i \in \mathbb{N}$. But before we can do so we need a better handle on

the projections $p_i \colon n \cdot Y \to Y$ in $\mathcal{K}\ell(T)$, for $i < n$. They are given inductively by:

$$(7)\quad p_0 = [\eta, \bot] \colon Y + n \cdot Y \longrightarrow T(Y) \quad \text{and} \quad p_{i+1} = [\bot, p_i] \colon Y + n \cdot Y \longrightarrow T(Y)$$

Then it is not hard to see that $p_i \circ \lambda_n = p_{n-i-1} \colon n \cdot Y \to T(Y)$, for $i < n$, and $p_i \circ \lambda_n = \bot$, for $i \geq n$.

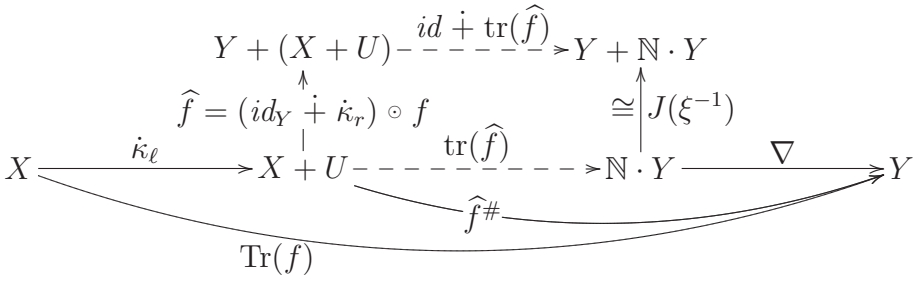
Next we use the explicit description of $\mathrm{tr}(c)$ as directed join from (5):

$$\begin{aligned} p_i \odot \mathrm{tr}(c) &= p_i \odot \Big(\bigvee\nolimits_{n \in \mathbb{N}} J(\lambda_n) \odot c_n \Big) \\ &= \bigvee\nolimits_{n \in \mathbb{N}} p_i \odot J(\lambda_n) \odot c_n \\ &= \bigvee\nolimits_{n \in \mathbb{N}} p_{n-i-1} \odot c_n \qquad \text{as we have just seen, where } i < n \\ &\overset{(*)}{=} c_\ell \odot c_r^i. \end{aligned}$$

The equation $(*)$ is obtained by induction on n, using (4). □

The main result of this paper now shows how coalgebraic traces in Kleisli categories yield a traced monoidal structure with respect to this monoidal structure $(0, +)$. The result is actually a direct consequence of Proposition 4.8, using [9, Theorem 3.1.4] (which dualises Hasegawa's result that uniform fixed point operators are uniform traces [11]). We should point out that the induced trace structure is of a very special kind, since the monoidal structure consists of coproducts, and the obtained trace operators are uniform. Hence it can equivalently be presented in terms of iteration operators à la Bloom-Ésik, *i.e.* as the duals of uniform fixed point operators, see [6]. So we are basically looking at an instance of Elgot iterative theories, see [4].

Theorem 5.2 *For $\mathbb{C}$ and T satisfying Requirements 4.7, the Kleisli category $\mathcal{K}\ell(T)$ with $(0, +)$ is traced monoidal (see [13]). For a map $f \colon X + U \to Y + U$ in $\mathcal{K}\ell(T)$ we define $\mathrm{Tr}(f) \colon X \to Y$ as the composite $\nabla \odot \mathrm{tr}(\widehat{f}) \odot \kappa_\ell = \widehat{f}^{\#} \odot \kappa_\ell$ at the bottom in:*



This monoidal trace operation Tr *then satisfies standard requirements from [13], and also the following special properties.*

Identity $\mathrm{Tr}(id_{X+U}) = id_X$;

Uniformity $\mathrm{Tr}(f) = \mathrm{Tr}(g)$, *if* $(id \dot{+} h) \odot f = g \odot (id \dot{+} h)$, *for* $f\colon X + U \to Y + U$, $g\colon X + V \to Y + V$ *and* $h\colon U \to V$ *(see [11]).*

Proof The result follows from the properties of iteration $(-)^{\#}$, see [9] [2], once we know that the definition of trace in [9] coincides with the coalgebraic one described in the theorem. This follows from Lemma 5.1 using a matrix description of $f\colon X + U \to Y + U$. Write $f_{ij} = \pi_j \odot f \odot \dot{\kappa}_i$, for $i, j \in \{\ell, r\}$, so that:

$$f = \begin{pmatrix} X \xrightarrow{f_{\ell\ell}} T(Y) & U \xrightarrow{f_{\ell r}} T(Y) \\ X \xrightarrow[f_{r\ell}]{} T(U) & U \xrightarrow[f_{rr}]{} T(U) \end{pmatrix}$$

We have to show that $\mathrm{Tr}(f) = \widehat{f}^{\#} \odot \dot{\kappa}_\ell$ as defined above can be written as the (regular) expression $f_{\ell\ell} \amalg f_{\ell r} f_{rr}^{\star} f_{r\ell}$ that is used in [9], and called the execution (or trace) formula. This follows from the description of iteration $(-)^{\#}$ in Lemma 5.1:

$$\begin{aligned}
\mathrm{Tr}(f) \;=\; \widehat{f}^{\#} \odot \dot{\kappa}_\ell \;&=\; p_\ell \odot \widehat{f} \odot \Big(\amalg_n \; (p_r \odot \widehat{f})^n \Big) \odot \dot{\kappa}_\ell \\
&= p_\ell \odot f \odot \Big(\amalg_n \; (\dot{\kappa}_r \odot p_r \odot f)^n \Big) \odot \dot{\kappa}_\ell \\
&= p_\ell \odot f \odot \Big(\mathrm{id} \amalg \amalg_n (\dot{\kappa}_r \odot p_r \odot f)^{n+1} \Big) \odot \dot{\kappa}_\ell \\
&= (p_\ell \odot f \odot \dot{\kappa}_\ell) \amalg (p_\ell \odot f \odot \Big(\amalg_n \; (\dot{\kappa}_r \odot p_r \odot f)^{n+1} \Big) \odot \dot{\kappa}_\ell) \\
&\overset{(*)}{=} f_{\ell\ell} \amalg (p_\ell \odot f \odot \Big(\amalg_n \; \dot{\kappa}_r \odot (p_r \odot f \odot \dot{\kappa}_r)^n \Big) \odot p_r \odot f \odot \dot{\kappa}_\ell) \\
&= f_{\ell\ell} \amalg \Big(p_\ell \odot f \odot \dot{\kappa}_r \odot \Big(\amalg_n \; (p_r \odot f \odot \dot{\kappa}_r)^n \Big) \odot f_{r\ell} \Big) \\
&= f_{\ell\ell} \amalg f_{\ell r} f_{rr}^{\star} f_{r\ell}.
\end{aligned}$$

The marked equation holds because

$$(\dot{\kappa}_r \odot p_r \odot f)^{n+1} \odot \dot{\kappa}_\ell = \dot{\kappa}_r \odot (p_r \odot f \odot \dot{\kappa}_r)^n \odot p_r \odot f \odot \dot{\kappa}_\ell,$$

which is obtained by induction.

The identity and uniformity properties are a consequence of Lemma 3.3. □

Example 5.3 We shall quickly review what this monoidal trace amounts to for a map $f\colon X + U \to T(Y + U)$ where T is one of the monads $\mathcal{P}, \mathcal{L}, \mathcal{D}, Q^{(-)}$ from Example 3.2.

[2] which, in dual form for products and a fixed point operator, should also be attributed to Masahito Hasegawa [11] and to Martin Hyland, see also [15].

(i) For the powerset monad $\mathcal{P}$ we get $\mathrm{Tr}(f)\colon X \to \mathcal{P}(Y)$ given by:

$$\begin{aligned} y \in \mathrm{Tr}(f)(x) &\iff \exists n \in \mathbb{N}.\, (n, y) \in \mathrm{tr}(f)(x) \\ &\iff \exists u_1, \ldots, u_n \in U.\, u_1 \in f(x) \wedge u_2 \in f(u_1) \wedge \cdots \\ &\qquad \wedge\, u_n \in f(u_{n-1}) \wedge y \in f(u_n). \end{aligned}$$

(ii) The lift monad yields $\mathrm{Tr}(f)\colon X \to 1 + Y$ as

$$\begin{aligned} \mathrm{Tr}(f)(x) = \mathrm{up}(y) \iff \exists n \in \mathbb{N}.\, f(x) = \mathrm{up}(u_1) \wedge f(u_1) = \mathrm{up}(u_2) \wedge \cdots \wedge \\ f(u_{n-1}) = \mathrm{up}(u_n) \wedge f(u_n) = \mathrm{up}(y). \end{aligned}$$

(iii) The subdistribution monad yields $\mathrm{Tr}(f)\colon X \to \mathcal{D}(Y)$ with:

$$\begin{aligned} &\mathrm{Tr}(f)(x)(y) \\ &= \sum_{n \in \mathbb{N}} \sum_{u_1, \ldots, u_n \in U} f(x)(u_1) \cdot f(u_1)(u_2) \cdot \ldots \cdot f(u_{n-1})(u_n) \cdot f(u_n)(y). \end{aligned}$$

(iv) Similarly, the quantale monad yields $\mathrm{Tr}(f)\colon X \to Q^Y$ with:

$$\begin{aligned} &\mathrm{Tr}(f)(x)(y) \\ &= \bigvee_{n \in \mathbb{N}} \bigvee_{u_1, \ldots, u_n \in U} f(x)(u_1) \cdot f(u_1)(u_2) \cdot \ldots \cdot f(u_{n-1})(u_n) \cdot f(u_n)(y). \end{aligned}$$

We have already seen that Kleisli categories of quantale monads are special, because they have biproducts. But there is more.

Lemma 5.4 *The Kleisli category $\mathcal{K}\ell(Q^{(-)})$ of the monad $Q^{(-)}$ for a commutative quantale Q has an involution $(-)^\dagger\colon \mathcal{K}\ell(Q^{(-)})^{op} \xrightarrow{\cong} \mathcal{K}\ell(Q^{(-)})$ that preserves biproducts and (monoidal) traces.*

Proof On objects one has $X^\dagger = X$ and on a morphism $f\colon X \to Q^Y$ one gets $f^\dagger\colon Y \to Q^X$ by $f^\dagger(y)(x) = f(x)(y)$. Clearly, $(-)^{\dagger\dagger} = \mathrm{id}$. Commutativity of Q's monoid $(1, \cdot)$ is needed to show that $(-)^\dagger$ preserves composition and traces. □

6 A category for bidirectional monadic computation

In this section we continue to work with a monad T on a category $\mathbb{C}$ as in Requirements 4.7 for which we thus have both coalgebraic traces (as in Proposition 3.1) and monoidal traces (by Theorem 5.2). Then we can apply the standard "Int" construction from [13]. We shall write $\mathcal{B}d(T)$ for the resulting category $\mathrm{Int}(\mathcal{K}\ell(T))$ of "bidirectional computations of type T".

This final section only contains an explicit description of this category $\mathcal{B}d(T)$ and a brief examination of our standard examples.

Definition 6.1 Let $\mathcal{B}d(T)$ be the category with:

Objects $A = (A_\ell, A_r)$ consisting of pairs of objects $A_\ell, A_r \in \mathbb{C}$;

Morphisms $f\colon A \to B$ are maps $f\colon A_\ell + B_r \to T(B_\ell + A_r)$ in $\mathbb{C}$. Of course they may also be described as maps $A_\ell + B_r \to B_\ell + A_r$ in the Kleisli category $\mathcal{K}\ell(T)$;

Identities $\mathrm{id}_A\colon A \to A$ are (Kleisli) identities $A_\ell + A_r \to T(A_\ell + A_r)$;

Composition For $f\colon A \to B$ and $g\colon B \to C$, that is for $f\colon A_\ell + B_r \to T(B_\ell + A_r)$ and $g\colon B_\ell + C_r \to T(C_\ell + B_r)$, the composite $g \circ f$ is the (monoidal) trace of the following "obvious" map: $(A_\ell + C_r) + B_r \to T((C_\ell + A_r) + B_r)$, given explicitly in $\mathcal{K}\ell(T)$ as:

$$\Big[\, \Big[[(\dot\kappa_1 \dotplus \mathrm{id}) \circ g \circ \dot\kappa_1, \dot\kappa_1 \circ \dot\kappa_2] \circ f \circ \dot\kappa_1,\ (\dot\kappa_1 \dotplus \mathrm{id}) \circ g \circ \dot\kappa_2 \Big] \\ [(\dot\kappa_1 \dotplus \mathrm{id}) \circ g \circ \dot\kappa_1, \dot\kappa_1 \circ \dot\kappa_2] \circ f \circ \dot\kappa_2 \Big].$$

We refer to [13] for the proof of the fact that this yields a compact closed category, with a full and faithful functor $\mathcal{K}\ell(T) \to \mathcal{B}d(T)$ given by $A \mapsto (A, 0)$. Such proofs are non-trivial, and can best be done using a suitable graphical notation.

In the remainder we briefly review our running examples. For the lift monad $\mathcal{L}$ the category $\mathcal{B}d(\mathcal{L})$ contains the essence of the category of games $\mathcal{G}$ as described in [3]. There, the objects can be described in terms of pairs of sets (A_ℓ, A_r) of moves, of a player (left, say) and opponent (right), together with additional structure, given by a set of plays, as suitable subset of the set of $(A_\ell + A_r)^\star$ sequences of moves. Morphisms $A \to B$ in $\mathcal{G}$ are "strategies", that can be described as certain partial functions $A_\ell + B_r \rightharpoonup B_\ell + A_r$, that is [3], as Kleisli maps $A_\ell + B_r \to 1 + (B_\ell + A_r)$. Composition of these strategies takes place via Girard's "execution formula", which corresponds to composition as described in Definition 6.1.

The category $\mathcal{B}d(\mathcal{D})$ for the distribution monad $\mathcal{D}$ does not seem to have been studied yet. The other example involving quantale monads gives rise to a separate result, yielding a setting for quantum computation, see [2]. It includes the familiar situation of relations.

Proposition 6.2 *The category $\mathcal{B}d(Q^{(-)})$ obtained from the quantale monad $Q^{(-)}$ for a commutative quantale Q is strongly compact closed.*

Proof The involution $(-)^\dagger$ from Lemma 5.4 is preserved by the "Int" construction, as claimed in [1]. □

[3] These strategies are maps $f\colon M_A^P + M_B^O \rightharpoonup M_A^O + M_B^P$ in the notation of [3, Section 2.4].

Acknowledgement

Thanks to are due to Masahito Hasegawa for helpful comments, and to referees of (earlier versions of) this paper.

References

[1] S. Abramsky. Abstract scalars, loops, and free traced and strongly compact closed categories. In J.L. Fiadeiro, N. Harman, M. Roggenbach, and J. Rutten, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2005)*, number 3629 in Lect. Notes Comp. Sci., pages 1–31. Springer, Berlin, 2005.

[2] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Logic in Computer Science*, pages 415–425. IEEE, Computer Science Press, 2004.

[3] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Inf. & Comp.*, 163:409–470, 2000.

[4] J. Adámek, S. Milius, and J. Velebil. Elgot theories: A new perspective of iteration theories (extended abstract). In S. Abramsky, M. Mislove, and C. Palamidessi, editors, *Mathematical Foundations of Programming Semantics*, number 249 in Elect. Notes in Theor. Comp. Sci., pages 407–427. Elsevier, Amsterdam, 2009.

[5] M.A. Arbib and E.G. Manes. *Algebraic Approaches to Program Semantics*. Texts and Monogr. in Comp. Sci. Springer, Berlin, 1986.

[6] S.L. Bloom and Z. Ésik. *Iteration Theories: The Equational Logic of Iterative Processes*. EATCS Monographs. Springer, Berlin, 1993.

[7] D. Coumans and B. Jacobs. Scalars, monads and categories, 2010. Manuscript.

[8] C.C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherson, editors, *Logic Colloquium '73*, pages 175–230, Amsterdam, 1975. North-Holland.

[9] E. Haghverdi. *A categorical approach to linear logic, geometry of proofs and full completeness*. PhD thesis, Univ. of Ottawa, Canada, 2000.

[10] E. Haghverdi. Unique decomposition categories, geometry of interaction and combinatory logic. *Math. Struct. in Comp. Sci.*, 10:205–231, 2000.

[11] M. Hasegawa. The uniformity principle on traced monoidal categories. In *Category Theory and Computer Science*, number 69 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2003.

[12] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace theory. *Logical Methods in Comp. Sci.*, 3(4:11), 2007.

[13] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Math. Proc. Cambridge Phil. Soc.*, 119(3):425–446, 1996.

[14] K.I. Rosenthal. *Quantales and their applications*. Number 234 in Pitman Research Notes in Math. Longman Scientific & Technical, 1990.

[15] A. Simpson and G. Plotkin. Complete axioms for categorical fixed-point operators. In *Logic in Computer Science*, pages 30–41. IEEE, Computer Science Press, 2000.

[16] M.B. Smyth and G.D. Plotkin. The category theoretic solution of recursive domain equations. *SIAM Journ. Comput.*, 11:761–783, 1982.

Higher-order Algebras and Coalgebras from Parameterized Endofunctors

Jiho Kim[1]

Department of Mathematics
Indiana University
Bloomington, Indiana, USA

Abstract

The study of algebras and coalgebras involve parametric description of a family of endofunctors. Such descriptions can often be packaged as *parameterized endofunctors*. A parameterized endofunctor generates a higher-order endofunctor on a functor category. We characterize initial algebras and final coalgebras for these higher-order endofunctors, generalizing several results in the literature.

Keywords: higher-order, algebras, coalgebras, parametric endofunctor

1 Introduction

Often, families of endofunctors with interesting algebras and coalgebras are defined by first fixing some parameters. More specifically, the definitions of endofunctors are usually related by having the same (multi-ary) functorial form. For instance, stream coalgebras arise from the bifunctor $\times\colon \mathsf{Set}\times\mathsf{Set}\to\mathsf{Set}$, where the first coordinate is fixed to be a particular set. This paper follows the lead of Kurz and Pattinson [7] and unify these definitions in the notion of a *parameterized endofunctor*.

A parameterized endofunctor generates a higher-order endofunctor on a functor category. For two categories $\mathcal{C}$ and $\mathcal{D}$, the functor category $[\mathcal{C},\mathcal{D}]$ consists of functors from $\mathcal{C}$ to $\mathcal{D}$ as objects and natural transformation among them as morphisms. While we treat the most general case, the two cases of particular interest in this paper are the category of endofunctors $\mathrm{End}(\mathcal{C}) = [\mathcal{C},\mathcal{C}]$ and the arrow category $\mathcal{C}^{\to}\cong[\mathbf{2},\mathcal{C}]$.

[1] Email: jihokim@indiana.edu

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

The main result of this paper is to characterize when such a construction will yield higher-order initial algebras and final coalgebras. The result is inspired by work done with initial algebras on arrow categories by Chuang and Lin [5], and also by another restricted case pertaining to iteratable functors given by Aczel, Adámek, Milius, and Velebil [1]. More constrained notions of parameterized endofunctors are presented in the literature, e.g. actions [4] and parameterized monads [9,2]. The work here, however, follows a relatively unconstrained approach. Initial algebras for higher-order endofunctors have been used to model the semantics of dependent types [5] and generalized algebraic data types (GADT's) [6]. Coalgebraically, higher-order endofunctors can be used to define higher-order, generic functions such as map on streams and other coinductive data-types.

The rest of the paper is organized in the following manner. Section 2 introduces the notion of parameterized endofunctors, making observations about some examples. Section 3 sets the stage for the main result by defining a certain completeness (and co-completeness) conditions on parameterized endofunctors which we call *suitability*. Section 4 states the main results and provides a detailed proof for the algebraic case. We also provide a sampling of how the theorems may applied in several disparate situations. We end with Section 5, providing some summarizing conclusions.

2 Functor categories and parameterized endofunctors

We begin with the definition of a parameterized endofunctor.

Definition 2.1 A $\mathcal{B}$-parameterized endofunctor on $\mathcal{C}$ is a bifunctor $F\colon \mathcal{B} \times \mathcal{C} \to \mathcal{C}$.

Alternatively, by the usual adjunction, the definition could be given as a functor from the parameter category $\mathcal{B}$ to the category of endofunctors $\mathrm{End}(\mathcal{C})$. While the description $\mathcal{B}$-*parameterized* becomes explicit in this modified form, the given definition will suffice for the sake of notational simplicity.

Given a parameterized endofunctor $F\colon \mathcal{B} \times \mathcal{C} \to \mathcal{C}$, every object $x \in \mathcal{B}$ restricts F to a $\mathcal{C}$-endofunctor which can be denoted as $F(x, \mathbb{1})\colon \mathcal{C} \to \mathcal{C}$. Moreover, for any morphism $x \xrightarrow{f} y$ in the parameter category, there is a natural transformation $F(x, \mathbb{1}) \xRightarrow{F(f,\mathbb{1})} F(y, \mathbb{1})$ given component-wise as $F(f, \mathbb{1})_c = F(f, c)$ for an object $c \in \mathcal{C}$.

There are many concrete examples of parameterized endofunctors, few of which are examined briefly here.

Example 2.2 For a non-empty set A, consider the Set-endofunctor $1 + A \times \mathbb{1}$. The initial $(1 + A \times \mathbb{1})$-algebra is A^*, the set of words on A. This endofunctor is "parameterized" by making A an argument to the bifunctor $F\colon \mathsf{Set} \times \mathsf{Set} \to \mathsf{Set}$

given by $F(A, X) = 1 + A \times X$ for $A, X \in \mathsf{Set}$.

Example 2.3 For non-empty sets A and B, consider the Set-functor $(B \times \mathbb{1})^A$. The $(B \times \mathbb{1})^A$-coalgebra $X \xrightarrow{f} (B \times X)^A$ corresponds to an automaton (X, A, B, f) where

- X is the state space,
- A and B are the sets of input and output symbols, respectively, and
- f determines the automaton's output and transition functions.

For a given state $x \in X$ and an input symbol $a \in A$, the output symbol $b \in \mathcal{B}$ and the next state $y \in X$ is given by the pair $\langle b, y\rangle = f(x)(a)$. Automata of this type are often called Mealy machines.

Let $\mathcal{B} = \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set}$ and $\mathcal{C} = \mathsf{Set}$. The parameterized endofunctor for this example is $F\colon (\mathsf{Set}^{\mathsf{op}} \times \mathsf{Set}) \times \mathsf{Set} \to \mathsf{Set}$, given by

$$F(\langle A, B\rangle, C) = (B \times C)^A.$$

for sets A, B, and C. F is contravariant in A and covariant in B (and C).

Example 2.4 Let $\mathbf{2} = \{0 \xleftarrow{!} 1\}$ be the 2-object category with a single non-identity morphism. For two endofunctors $G_0, G_1\colon \mathcal{C} \to \mathcal{C}$ and a natural transformation $G_1 \xRightarrow{\theta} G_0$, let $F\colon \mathbf{2} \times \mathcal{C} \to \mathcal{C}$ be the parameterized endofunctor given by

$$F(i, x) = G_i x \qquad\qquad F(!, x) = \theta_x$$

for $i \in \mathbf{2}$ and $x \in \mathcal{C}$. In short, F is the natural transformation θ.

Example 2.5 For a $\mathcal{C}$-endofunctor H and an object $c \in \mathcal{C}$ consider the $\mathcal{C}$-endofunctor $F_{H,c}$ given by $F_{H,c}(x) = c + Hx$. The corresponding parameterized endofunctor is $F\colon (\mathrm{End}(\mathcal{C}) \times \mathcal{C}) \times \mathcal{C} \to \mathcal{C}$ given by

$$F(\langle H, c\rangle, x) = c + Hx.$$

3 Suitability

Ultimately, the interest in parameterized endofunctors here is to consider their relationship to the theory of algebras and coalgebras. In this vein, we introduce the notion of suitability.

Definition 3.1 A $\mathcal{B}$-parameterized endofunctor $F\colon \mathcal{B} \times \mathcal{C} \to \mathcal{C}$ is *initially suitable* if for every object $x \in \mathcal{B}$, the endofunctor $F(x, \mathbb{1})\colon \mathcal{C} \to \mathcal{C}$ admits an initial algebra. Dually, F is *finally suitable* if for every object $x \in \mathcal{B}$, the endofunctor $F(x, \mathbb{1})$ admits a final coalgebra.

Suppose $F\colon \mathcal{B} \times \mathcal{C} \to \mathcal{C}$ is initially suitable. For each $x \in \mathcal{B}$, let $F(x, \mathcal{R}_F x) \xrightarrow{r_x} \mathcal{R}_F x$ be the initial $F(x, \mathbb{1})$-algebra. $\mathcal{R}_F$ extends to a functor $\mathcal{R}_F\colon \mathcal{B} \to \mathcal{C}$ by

mapping a $\mathcal{B}$-morphism $x \xrightarrow{f} y$ to the unique algebra morphism, denoted $\mathcal{R}_F f$, induced by initiality in the following commutative diagram:

$$\begin{array}{ccccc} F(x, \mathcal{R}_F x) & & \xrightarrow{\quad r_x \quad} & & \mathcal{R}_F x \\ {\scriptstyle F(x,\mathcal{R}_F f)}\downarrow & \cdots\cdots & {\scriptstyle F(f,\mathcal{R}_F f)=F(\mathbb{1},\mathcal{R}_F)f} & \cdots\searrow & \downarrow{\scriptstyle \mathcal{R}_F f} \\ F(x, \mathcal{R}_F y) & \xrightarrow[F(f,R_F y)]{} & F(y, \mathcal{R}_F y) & \xrightarrow[r_y]{} & \mathcal{R}_F y \end{array} \tag{1}$$

Dually, suppose F is finally suitable. Then for $x \in \mathcal{B}$, let $\mathcal{S}_F x \xrightarrow{s_x} F(x, \mathcal{S}_F x)$ be the final $F(x, \mathbb{1})$-coalgebra. $\mathcal{S}_F$ extends to a functor $\mathcal{S}_F \colon \mathcal{B} \to \mathcal{C}$ by mapping a $\mathcal{B}$-morphism $x \xrightarrow{f} y$ to the unique coalgebra morphism, denoted $\mathcal{S}_F f$, induced by finality in the following commuting diagram:

$$\begin{array}{ccccc} \mathcal{S}_F x & \xrightarrow{\quad s_x \quad} & F(x, \mathcal{S}_F x) & \xrightarrow{\quad F(f,\mathcal{S}_F x) \quad} & F(y, \mathcal{S}_F x) \\ {\scriptstyle \mathcal{S}_F f}\downarrow & & \cdots\cdots & {\scriptstyle F(\mathbb{1},\mathcal{S}_F)f=F(f,\mathcal{S}_F f)} \cdots\searrow & \downarrow{\scriptstyle F(y,\mathcal{S}_F f)} \\ \mathcal{S}_F y & & \xrightarrow[s_y]{} & & F(y, \mathcal{S}_F y) \end{array} \tag{2}$$

The structure morphisms from the initial algebras and final coalgebras above collectively form two natural transformations:

$$F(\mathbb{1}, \mathcal{R}_F) \xRightarrow{\quad r \quad} \mathcal{R}_F \qquad\qquad \mathcal{S}_F \xRightarrow{\quad s \quad} F(\mathbb{1}, \mathcal{S}_F) \tag{3}$$

The naturality condition is evidently satisfied through the dotted arrows in (1) and (2). Furthermore, both of these natural transformations are isomorphisms by Lambek's Lemma applied to each component.

The definition of initial and final suitability generalizes a collection of common concepts in the theory of algebras and coalgebras. Free monads, completely iterative monads, and their duals are in fact based on initial or final suitability conditions for certain parameterized endofunctors. The following examples clarify the nature of how initial and final suitability generalizes and unifies these notions.

Example 3.2 Given an endofunctor H on a category $\mathcal{C}$ with binary coproducts, we have the parameterized endofunctor $F \colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ given by

$$F(c, x) = c + Hx. \tag{4}$$

If F is initially suitable, then $\mathcal{R}_F$ is called the *free monad generated by* H [3]. If F is finally suitable, then H is called *iteratable*, and $\mathcal{S}_F$ is called the *completely iterative monad generated by* H [1].

Example 3.3 Given an endofunctor H on a category $\mathcal{C}$ with binary products, we have the parameterized endofunctor $F\colon \mathcal{C}\times\mathcal{C}\to\mathcal{C}$ given by

$$F(c,x) = c \times Hx. \tag{5}$$

If F is initially suitable, $\mathcal{R}_F$ is called *cofree recursive comonad generated by* H. If F is finally suitable, $\mathcal{S}_F$ is called the *cofree comonad generated by* H [10].

The monadic and comonadic structures in Examples 3.2 and 3.3 are artifacts of the particular shapes the parameterized endofunctors take in (4) and (5). $\mathcal{R}_F$ and $\mathcal{S}_F$ will not have an obvious monad or comonad structure in general.

For further examples, we elaborate on Examples 2.3 and 2.4.

Example 3.4 A stream function $A^\omega \xrightarrow{f} B^\omega$ is causal if it is non-expanding in the usual metric on streams given by

$$d(\sigma,\tau) = \begin{cases} 0 & \text{if } \sigma = \tau \\ 2^{-i} & \text{if } \sigma \neq \tau \end{cases}$$

where i is the length of the longest common prefix of σ and τ. Intuitively, two streams are close together if they share a long prefix. If two streams share a common prefix, then their images under a non-expansive function share a prefix of the same (or greater) length. For this reason, if f is non-expansive, $\mathsf{hd}(f(a{:}\sigma)) = \mathsf{hd}(f(a{:}\tau))$, regardless of the choice of σ and τ.

The final $(B\times\mathbb{1})^A$-coalgebra is carried by the set $\Gamma_{A,B}$ of causal stream functions from A^ω to B^ω [8]. The structure map of the final $(B\times\mathbb{1})^A$-coalgebra $\Gamma_{A,B} \xrightarrow{\gamma_{A,B}} (B\times\Gamma_{A,B})^A$ is given by

$$\gamma_{A,B}(f)(a) = \langle \mathsf{hd}\circ f\circ c_a, \mathsf{tl}\circ f\circ c_a\rangle$$

Here c_a is the mapping $\sigma\mapsto a{:}\sigma$. (Recall also that for a set A, the pairing of the head and tail maps on streams, i.e. $A^\omega \xrightarrow{\langle \mathsf{hd},\mathsf{tl}\rangle} A\times A^\omega$, is the final $(A\times\mathbb{1})$-coalgebra.) Per the observation in the previous paragraph, $\mathsf{hd}\circ f\circ c_a$ is constant to B since f is causal. By abusing notation, the first coordinate of $\gamma_{A,B}(f)(a)$ is written as a function $A^\omega\to B$ for the sake of symmetry.

Example 3.5 The parameterized endofunctor F from Example 2.4 is initial suitable (resp. finally suitable) if both G_0 and G_1 admit initial algebras (resp. final coalgebras). If F is initially suitable, then let $G_i a_i \xrightarrow{r_i} a_i$ be the initial G_i-algebra carried by $a_i = \mathcal{R}_F i$ for $i\in\mathbf{2}$. By initiality of $G_1 a_1 \xrightarrow{r_1} a_1$,

there is a unique G_1-algebra morphism ζ so that

$$\begin{array}{ccccc} G_1a_1 & & \xrightarrow{\quad r_1 \quad} & & a_1 \\ {\scriptstyle F(1,\zeta)=G_1\zeta}\downarrow & & & & \downarrow{\scriptstyle \zeta} \\ G_1a_0 & \xrightarrow[F(!,a_0)]{\theta_{a_0}} & G_0a_0 & \xrightarrow{r_0} & a_0 \end{array} \tag{6}$$

commutes. In this case, the functor $\mathcal{R}_F \colon \mathbf{2} \to \mathcal{C}$ can be identified with the $\mathcal{C}$-morphism ζ.

4 Higher-order algebras and coalgebras

The study of algebras and coalgebras often depend heavily on the choice of the base category. Generally speaking, it is often fruitful to fix a category and consider interesting families of endofunctors, which either admit algebras or coalgebras or both. The proposal in this research is to consider functor categories as an appealing option for the fixed category.

In sequel, we refer to algebras and coalgebras defined via endofunctors on functor categories as higher-order algebras and coalgebras.

Any study of higher-order algebras and coalgebras are inevitably subsumed in the general theory since we are only fixing some particular class of categories to focus on. However the higher-order approach also extends the general theory in the following sense. Given any category $\mathcal{C}$, an endofunctor $F \colon \mathcal{C} \to \mathcal{C}$ can be embedded as an endofunctor on the functor category $[\mathbf{1}, \mathcal{C}]$, where $\mathbf{1}$ is the terminal category. By allowing different categories in the place of $\mathbf{1}$, richer structures may be discerned and utilized.

4.1 Higher-order endofunctor generated by a parameterized endofunctor

There is no doubt that characterizing higher-order endofunctors and their algebras and coalgebras in full generality is an insurmountably difficult task. We take a much more modest approach of investigating a particular class of higher-order endofunctors which arise naturally from parameterized endofunctors.

Definition 4.1 Let $F \colon \mathcal{B} \times \mathcal{C} \to \mathcal{C}$ be a parameterized endofunctor. Define an higher-order endofunctor $\widehat{F} \colon [\mathcal{B}, \mathcal{C}] \to [\mathcal{B}, \mathcal{C}]$ by $\widehat{F}X = F(\mathbb{1}_{\mathcal{B}}, X)$ for a functor $X \colon \mathcal{B} \to \mathcal{C}$. For a natural transformation $X \overset{\lambda}{\Rightarrow} Y$, the natural transformation $\widehat{F}\lambda$ is given component-wise by $F(\mathbb{1}, \lambda)_b = F(b, \lambda_b)$ for $b \in \mathcal{B}$.

We say $\widehat{F}$ is the *higher-order endofunctor generated by the parameterized endofunctor* F.

Example 4.2 Given an endofunctor $H \colon \mathcal{C} \to \mathcal{C}$, we can produce a higher-order endofunctor $H \circ \mathbb{1} \colon [\mathcal{B}, \mathcal{C}] \to [\mathcal{B}, \mathcal{C}]$ by post-composition. $H \circ \mathbb{1}$ can be

generated by parameterized endofunctor $F\colon \mathcal{B}\times\mathcal{C} \to \mathcal{C}$ given by $F(x,y) = Hy$. In this case, F is initially (resp. finally) suitable if and only if H admits an initial algebra (resp. a final coalgebra).

Example 4.3 Given an endofunctor $G\colon \mathcal{B} \to \mathcal{B}$, we can produce a higher-order endofunctor $\mathbb{1}\circ G\colon [\mathcal{B},\mathcal{C}] \to [\mathcal{B},\mathcal{C}]$ by pre-composition. This higher-order endofunctor cannot be generated by a parameterized endofunctor in general.

Example 4.4 We continue here with Example 3.5. The parameterized endofunctor $F\colon \mathbf{2}\times\mathcal{C} \to \mathcal{C}$ generates an endofunctor $\widehat{F}$ on the arrow category $\mathcal{C}^{\to} \cong [\mathbf{2},\mathcal{C}]$. Objects of $\mathcal{C}^{\to}$ are $\mathcal{C}$-morphisms. A $\mathcal{C}^{\to}$-morphism from $x \xrightarrow{a} y$ to $x' \xrightarrow{a'} y'$ is a pair of $\mathcal{C}$-morphisms $m = \langle x \xrightarrow{m_x} x', y \xrightarrow{m_y} y'\rangle$ so that the square

$$\begin{array}{ccc} x & \xrightarrow{\ a\ } & y \\ {\scriptstyle m_x}\downarrow & & \downarrow{\scriptstyle m_y} \\ x' & \xrightarrow[\ a'\ ]{} & y' \end{array}$$

commutes.

The image of a $\mathcal{C}^{\to}$-object $x \xrightarrow{a} y$ under $\widehat{F}$ is the composition

$$G_1x \xrightarrow[F(1,a)]{G_1a} G_1y \xrightarrow[F(!,y)]{\theta_y} G_0y \quad \text{or} \quad G_1x \xrightarrow[F(!,x)]{\theta_x} G_0x \xrightarrow[F(0,a)]{G_0a} G_0y.$$

which are equal by the naturality of θ. For a $\mathcal{C}^{\to}$-morphism $m = \langle x \xrightarrow{m_x} x', y \xrightarrow{m_y} y'\rangle$, we have $\widehat{F}m = \langle G_1m_x, G_0m_y\rangle$.

4.2 Algebra and coalgebra of higher-order endofunctors

In this section we discuss the necessary and sufficient conditions for higher-order endofunctors generated by parameterized endofunctors to admit initial algebras or final coalgebras.

As noted earlier, for a parameterized endofunctor $F\colon \mathcal{B}\times\mathcal{C} \to \mathcal{C}$, which is initially (resp. finally) suitable, there is a natural transformation $F(\mathbb{1},\mathcal{R}_F) \overset{r}{\Rightarrow} \mathcal{R}_F$ (resp. $\mathcal{S}_F \overset{s}{\Rightarrow} F(\mathbb{1},\mathcal{S}_F)$). In the context of the higher-order endofunctor $\widehat{F}$ generated by F, the natural transformation r is an $\widehat{F}$-algebra and s is a $\widehat{F}$-coalgebra:

$$\widehat{F}\mathcal{R}_F \xRightarrow{\ r\ } \mathcal{R}_F \qquad\qquad \mathcal{S}_F \xRightarrow{\ s\ } \widehat{F}\mathcal{S}_F$$

When F is initially suitable, $(\mathcal{R}_F, r)$ will be the initial $\widehat{F}$-algebra, and dually when F is finally suitable, $(\mathcal{S}_F, s)$ will be the final $\widehat{F}$-coalgebra.

Theorem 4.5 *Let $\mathcal{C}$ be a locally small category with powers. For a higher-order endofunctor $\widehat{F}\colon [\mathcal{B},\mathcal{C}] \to [\mathcal{B},\mathcal{C}]$ generated by a parameterized endofunctor $F\colon \mathcal{B}\times\mathcal{C} \to \mathcal{C}$, the following are equivalent:*

(i) F *is initially suitable.*

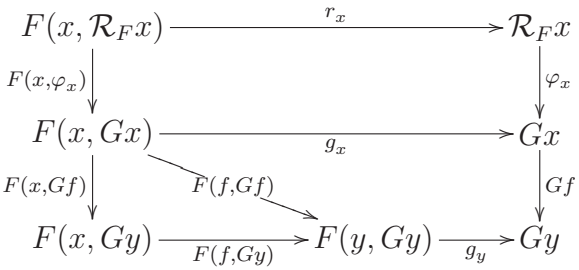
(ii) $\widehat{F}$ *admits an initial algebra.*

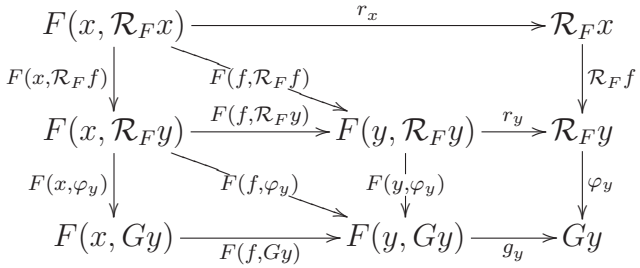
In fact, given an initially suitable F*, the initial algebra of* $\widehat{F}$ *is* $\widehat{F}\mathcal{R}_F \overset{r}{\Rightarrow} \mathcal{R}_F$*. Conversely, given an object* $x \in \mathcal{B}$ *and an initial* $\widehat{F}$*-algebra* $\widehat{F}A \overset{\alpha}{\Rightarrow} A$*, the initial* $F(x, \mathbb{1})$*-algebra is just* $(\widehat{F}A)x = F(x, Ax) \xrightarrow{\alpha_x} Ax$*.*

Proof. For (i)$\Longrightarrow$(ii), suppose F is initially suitable. We will show that $(\mathcal{R}_F, r)$ is an initial $\widehat{F}$-algebra. To that end, let $\widehat{F}G \overset{g}{\Rightarrow} G$ be an $\widehat{F}$-algebra. For every $x \in \mathcal{B}$, there exists a unique $F(x, \mathbb{1})$-algebra morphism, $\mathcal{R}_F x \xrightarrow{\varphi_x} Gx$, making the square

$$\begin{array}{ccc} F(x, \mathcal{R}_F x) & \xrightarrow{r_x} & \mathcal{R}_F x \\ {\scriptstyle F(x,\varphi_x)}\downarrow & & \downarrow{\scriptstyle \varphi_x} \\ F(x, Gx) & \xrightarrow[g_x]{} & Gx \end{array} \tag{7}$$

commute because r_x is the initial $F(x, \mathbb{1})$-algebra. We need to show that φ is natural. For a morphism $x \xrightarrow{f} y$, consider the following diagrams:





The triangles all commute trivially. The squares commute by definition of φ (7), and the trapezoids commute because both g and r are natural. These diagrams above show that $Gf \circ \varphi_x$ and $\varphi_y \circ \mathcal{R}_F f$ are both $F(x, \mathbb{1})$-algebra morphisms from an initial algebra r_x to the algebra $g_y \circ F(f, Gy)$. By initiality these morphisms must be equal, showing that φ is indeed natural. The uniqueness of φ as an $\widehat{F}$-algebra morphism follow directly from the uniqueness of each component of φ as an $F(x, \mathbb{1})$-algebra morphism.

Conversely, for (ii)$\Longrightarrow$(i), suppose $\widehat{F}$ admits an initial algebra. Then there

is a functor $A\colon \mathcal{B} \to \mathcal{C}$ and a natural transformation

$$F(\mathbb{1}, A) = \widehat{F}A \xRightarrow{\quad\alpha\quad} A$$

so that (A, α) is initial among all $\widehat{F}$-algebras. We will demonstrate that $F(x, Ax) \xrightarrow{\alpha_x} Ax$ is an initial $F(x, \mathbb{1})$-algebra.

For $x \in \mathcal{B}$ and $y \in \mathcal{C}$, we define a functor $J_{x,y}\colon \mathcal{B} \to \mathcal{C}$ given by $J_{x,y}a = \prod_{\mathcal{B}(a,x)} y$ for $a \in \mathcal{C}$. Given a $\mathcal{B}$-morphism $a \xrightarrow{f} b$, the $\mathcal{C}$-morphism $\prod_{\mathcal{B}(a,x)} y \xrightarrow{J_{x,y}f} \prod_{\mathcal{B}(b,x)} y$ is given by $J_{x,y}f = \langle \pi_{g\circ f} \rangle_{g \in \mathcal{B}(b,x)}$, or equivalently,

$$\pi_g \circ J_{x,y}f = \pi_{g\circ f} \tag{8}$$

for $g \in \mathcal{B}(b, x)$. For any functor $S\colon \mathcal{B} \to \mathcal{C}$ parallel to $J_{x,y}$, there is a bijective correspondence

$$\mathrm{Nat}(S, J_{x,y}) \underset{(-)^\sharp}{\overset{(-)^\flat}{\rightleftarrows}} \mathrm{Hom}(Sx, y) \tag{9}$$

(From a broader perspective, this bijective correspondence is the consequence of $J_{x,y}$ being the right Kan extension $\mathrm{Ran}_X Y$ of the functor $Y\colon \mathbf{1} \to \mathcal{C}$ along $X\colon \mathbf{1} \to \mathcal{B}$ which are constant on $y \in \mathcal{C}$ and $x \in \mathcal{B}$ respectively.) For a natural transformation $S \overset{\lambda}{\Longrightarrow} J_{x,y}$, the $\mathcal{C}$-morphism $Sx \xrightarrow{\lambda^\flat} y$ is given by the composition

$$Sx \xrightarrow{\quad\lambda_x\quad} J_{x,y}x = \prod_{\mathcal{B}(x,x)} y \xrightarrow{\quad\pi_{\mathsf{id}_x}\quad} y. \tag{10}$$

Conversely, given a morphism $Sx \xrightarrow{u} y$, the components of the natural transformation $S \overset{u^\sharp}{\Longrightarrow} J_{x,y}$ is given by

$$u^\sharp_b = \langle u \circ Sg \rangle_{g \in \mathcal{B}(b,x)} \tag{11}$$

$$\pi_g \circ u^\sharp_b = u \circ Sg \tag{12}$$

for $g \in \mathcal{B}(b, x)$. We can see that

$$(u^\sharp)^\flat \overset{(10)}{=} \pi_{\mathsf{id}_x} \circ u^\sharp_x \overset{(11)}{=} \pi_{\mathsf{id}_x} \circ \langle u \circ Sg \rangle_{g \in \mathcal{B}(x,x)} = u \circ S(\mathsf{id}_x) = u \tag{13}$$

and for $b \in \mathcal{B}$,

$$\begin{aligned}(\lambda^\flat)^\sharp_b &\overset{(11)}{=} \langle \lambda^\flat \circ Sg \rangle_{g \in \mathcal{B}(b,x)} \overset{(10)}{=} \langle \pi_{\mathsf{id}_x} \circ \lambda_x \circ Sg \rangle_{g \in \mathcal{B}(b,x)} \\ &\overset{(*)}{=} \langle \pi_{\mathsf{id}_x} \circ J_{x,y}g \circ \lambda_b \rangle_{g \in \mathcal{B}(b,x)} \\ &\overset{(8)}{=} \langle \pi_g \circ \lambda_b \rangle_{g \in \mathcal{B}(b,x)} = \langle \pi_g \rangle_{g \in \mathcal{B}(b,x)} \circ \lambda_b = \lambda_b.\end{aligned}$$

The equality marked $(*)$ is due to the naturality of λ.

Let $F(x,y) \xrightarrow{u} y$ be an arbitrary $F(x,\mathbb{1})$-algebra. Composing with $F(x,\pi_{\mathsf{id}_x})$, we have

$$F(\mathbb{1},J_{x,y})(x) = F(x,J_{x,y}x) = F(x,\textstyle\prod_{\mathcal{B}(x,x)} y) \xrightarrow{F(x,\pi_{\mathsf{id}_x})} F(x,y) \xrightarrow{u} y$$

which is of the form $Sx \to y$, for the functor $S = F(\mathbb{1},J_{x,y})$. By the bijective correspondence (9), we obtain an $\widehat{F}$-algebra

$$F(\mathbb{1},J_{x,y}) = \widehat{F}J_{x,y} \xRightarrow{(u\circ F(x,\pi_{\mathsf{id}_x}))^\sharp} J_{x,y}\,,$$

Then, we have an $\widehat{F}$-algebra morphism $A \xRightarrow{\psi} J_{x,y}$ so that the diagram

$$\begin{array}{ccc} F(\mathbb{1},A) & \xRightarrow{\alpha} & A \\ {\scriptstyle F(\mathsf{id},\psi)}\Downarrow & & \Downarrow{\scriptstyle \psi} \\ F(\mathbb{1},J_{x,y}) & \xRightarrow[(u\circ F(x,\pi_{\mathsf{id}_x}))^\sharp]{} & J_{x,y} \end{array} \tag{14}$$

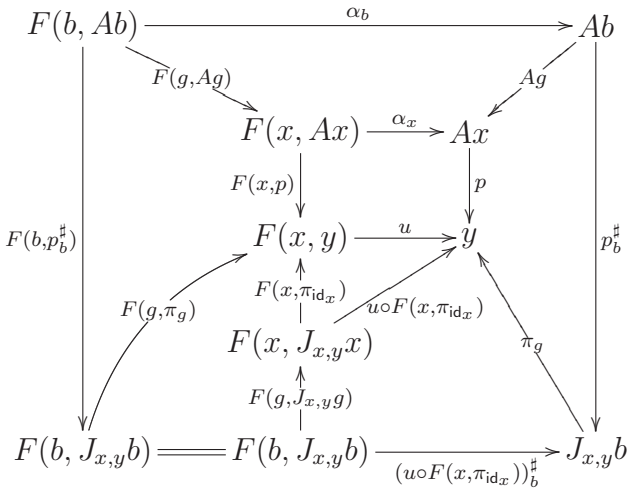
commutes by the initiality of (A,α). Note that the natural transformation ψ here depends on u. Recalling that $\psi^\flat = \pi_{\mathsf{id}_x} \circ \psi_x$ (10), consider the following commutative diagram:

$$\begin{array}{ccccc} & F(x,Ax) & \xrightarrow{\alpha_x} & Ax & \\ & \downarrow{\scriptstyle F(x,\psi_x)} & & \downarrow{\scriptstyle \psi_x} & \\ {\scriptstyle F(x,\psi^\flat)}\quad F(x,J_{x,y}x) & & \xrightarrow{(u\circ F(x,\pi_{\mathsf{id}_x}))^\sharp_x} & J_{x,y}x & \quad{\scriptstyle \psi^\flat} \\ & \downarrow{\scriptstyle F(x,\pi_{\mathsf{id}_x})} & & \downarrow{\scriptstyle \pi_{\mathsf{id}_x}} & \\ & F(x,y) & \xrightarrow[u]{} & y & \end{array}$$

The top square commutes due to the initiality of α (14), and the bottom square is the identity $f = \pi_{\mathsf{id}_x} \circ f^\sharp_x$ (13), where $f = u \circ F(x,\pi_{\mathsf{id}_x})$. Therefore, $\psi^\flat$ is an $F(x,\mathbb{1})$-algebra morphism.

Next, suppose $Ax \xrightarrow{p} y$ is an $F(x,\mathbb{1})$-algebra morphism from $F(x,Ax) \xrightarrow{\alpha_x} Ax$ to $F(x,y) \xrightarrow{u} y$. For uniqueness, we must verify that $\psi^\flat = p$. To that end,

consider the following diagram.



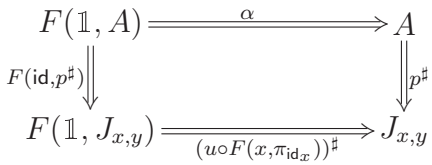
Here g is an arbitrary morphism in $\mathcal{B}(b, x)$. The center square commutes by assumption that p is an algebra morphism. The region above it commutes by naturality of α; the region to the right is an instance of (12); the region to the left consequently commutes by bifunctoriality of F. The triangle below the center square commutes trivially. The region below the triangle is another instance of (12), because the arrow

$$F(b, J_{x,y}b) \xrightarrow{F(g,J_{x,y}g)} F(x, J_{x,y}x)$$

is just $Sb \xrightarrow{Sg} Sx$ for $S = F(\mathbb{1}, J_{x,y})$. Finally, the region to the left of the triangle commutes by the definition of $J_{x,y}$ on morphisms (8). Therefore, for any $b \in \mathcal{B}$ and $g \in \mathcal{B}(b, x)$:

$$\begin{aligned}
\pi_g \circ (p^\sharp \circ \alpha)_b &= \pi_g \circ \left[p_b^\sharp \circ \alpha_b\right] \\
&= \pi_g \circ \left[(u \circ F(x, \pi_{\mathsf{id}_x}))_b^\sharp \circ F(b, p_b^\sharp)\right] \\
&= \pi_g \circ ((u \circ F(x, \pi_{\mathsf{id}_x}))^\sharp \circ F(\mathsf{id}, p^\sharp))_b.
\end{aligned}$$

These calculations show that $(p^\sharp \circ \alpha)_b = ((u \circ F(x, \pi_{\mathsf{id}_x}))^\sharp \circ F(\mathsf{id}, p^\sharp))_b$, and consequently, that the following diagram of natural transformations commutes.



That is to say, $p^\sharp$ is an $\widehat{F}$-algebra morphism. By initiality, we conclude that

$p^\sharp = \psi$. Therefore, $\psi^\flat = (p^\sharp)^\flat \overset{(13)}{=} p$, as required for uniqueness of the $F(x, \mathbb{1})$-algebra morphism from (Ax, α_x) to any other $F(x, \mathbb{1})$-algebra. □

Theorem 4.6 *Let $\mathcal{C}$ be a locally small category with copowers. For a higher-order endofunctor $\widehat{F}\colon [\mathcal{B}, \mathcal{C}] \to [\mathcal{B}, \mathcal{C}]$ generated by a parameterized endofunctor $F\colon \mathcal{B} \times \mathcal{C} \to \mathcal{C}$, the following are equivalent:*

(i) F is finally suitable.

(ii) $\widehat{F}$ admits a final coalgebra.

Proof. Dualize the proof to the previous theorem. The bijective correspondence in this case

$$\mathrm{Nat}(K_{x,y}, Q) \rightleftarrows \mathrm{Hom}(y, Qx) \tag{15}$$

will come from the left Kan extension $\mathrm{Lan}_X Y = K_{x,y}$ which is formed by copowers:

$$K_{x,y}a = \coprod_{\mathcal{B}(x,b)} y.$$

The details can be gleaned from proof of Corollary 4.7 first proven in Aczel, Adámek, Milius, and Velebil [1]. □

Corollary 4.7 *For an endofunctor $H\colon \mathcal{C} \to \mathcal{C}$ on a locally small category $\mathcal{C}$ with copowers, the following are equivalent:*

(i) H is iteratable.

(ii) The higher-order endofunctor $\widehat{H}\colon [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$ given by $\widehat{H}X = \mathbb{1} + HX$ admits a final coalgebra.

Proof. Invoke Theorem 4.6 on the parameterized endofunctor $F\colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ given by $F(x, y) = x + Hy$. □

4.3 map *as a higher-order coalgebra morphism*

In this section, we continue Examples 2.3 and 3.4. Let $F\colon (\mathsf{Set}^{\mathsf{op}} \times \mathsf{Set}) \times \mathsf{Set} \to \mathsf{Set}$ be the parameterized endofunctor given by $F(\langle A, B\rangle, C) = (B \times C)^A$. It generates a higher-order endofunctor $\widehat{F}$ on $[\mathsf{Set}^{\mathsf{op}} \times \mathsf{Set}, \mathsf{Set}]$ so that

$$(\widehat{F}X)\langle A, B\rangle = (B \times X(A, B))^A$$

for a functor $X\colon \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$.

As noted in Example 3.4, F is finally suitable, and produces a functor $\Gamma = \mathcal{S}_F\colon \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$ which is given by $\Gamma\langle A, B\rangle = \Gamma_{A,B}$, the set of causal functions from A^ω to B^ω. Theorem 4.6 yields a final $\widehat{F}$-coalgebra

$$\Gamma \xrightarrow{\quad\gamma\quad} \widehat{F}\Gamma = F(\mathbb{1}, \Gamma)$$

given by $\gamma_{\langle A,B\rangle} = \gamma_{A,B}$.

Fix a functor $H\colon \mathsf{Set}^{\mathsf{op}} \times \mathsf{Set} \to \mathsf{Set}$, given by $H\langle A, B\rangle = \mathrm{Hom}(A, B) = B^A$. We define a higher-order $\widehat{F}$-coalgebra $H \overset{e}{\Rightarrow} F(\mathbb{1}, H)$ by specifying its components

$$e_{\langle A,B\rangle} : B^A \to (B \times B^A)^A$$

with $e_{\langle A,B\rangle}(f)(a) = \langle f(a), f\rangle$. Finality of the higher-order $\widehat{F}$-coalgebra (Γ, γ) produces an $\widehat{F}$-coalgebra morphism (i.e. a natural transformation) $H \overset{m}{\Longrightarrow} \Gamma$ so that

$$\gamma \circ m = F(\mathbb{1}, m) \circ e.$$

The function $m_{\langle A,B\rangle}\colon B^A \to \Gamma_{A,B}$ can be given as

$$m_{\langle A,B\rangle}(f)(\alpha_0, \alpha_1, \alpha_2, \ldots) = (f(\alpha_0), f(\alpha_1), f(\alpha_2), \ldots).$$

for $f\colon A \to B$ and $\alpha = (\alpha_0, \alpha_1, \alpha_2, \ldots) \in A^\omega$. More succinctly, $m_{\langle A,B\rangle}$ is more commonly known as map, the morphism mapping of the Set-endofunctor $\mathbb{1}^\omega$. Here we have derived map as a higher-order coalgebra morphism induced by the finality of (Γ, γ).

4.4 Algebras in arrow categories

In this section, we conclude the discussion of arrow categories from Examples 2.4, 3.5, and 4.4.

An $\widehat{F}$-algebra $u = \langle u_x, u_y\rangle$ and $\widehat{F}$-coalgebra $v = \langle v_x, v_y\rangle$ make the diagrams

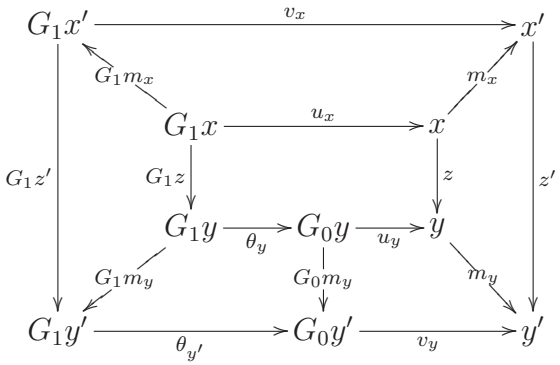
$$\begin{array}{ccccc} G_1x & \xrightarrow{G_1z} & G_1y & \xrightarrow{\theta_y} & G_0y \\ {\scriptstyle u_x}\downarrow & & & & \downarrow{\scriptstyle u_y} \\ x & & \xrightarrow[z]{} & & y \end{array} \qquad \begin{array}{ccccc} x & & \xrightarrow{z} & & y \\ {\scriptstyle v_x}\downarrow & & & & \downarrow{\scriptstyle v_y} \\ G_1x & \xrightarrow[\theta_x]{} & G_0x & \xrightarrow[G_0z]{} & G_0y \end{array} \tag{16}$$

commute. For the sake of brevity, we will only continue with the algebraic aspect; the coalgebraic perspective is completely parallel. Consider the diagrams for $\widehat{F}$-algebras (16). From another perspective, an $\widehat{F}$-algebra $\widehat{F}z \xrightarrow{u} z$ can be viewed as a G_1-algebra morphism f from u_x to $u_y \circ \theta_y$:

$$\begin{array}{ccccc} G_1x & & \xrightarrow{u_x} & & x \\ {\scriptstyle G_1z}\downarrow & & & & \downarrow{\scriptstyle z} \\ G_1y & \xrightarrow[\theta_y]{} & G_0y & \xrightarrow[u_y]{} & y \end{array} \tag{17}$$

An $\widehat{F}$-algebra morphism from (z, u) to (z', v) is a pair of $\mathcal{C}$-morphisms $m =$

$\langle m_x, m_y \rangle$ so that



(18)

commutes. This diagram can be characterized by the following facts:

(i) (z, u) and (z', v) are $\widehat{F}$-algebras.

(ii) The pair $m = \langle m_x, m_y \rangle$ is a $\mathcal{C}^{\to}$-morphism from z to z'.

(iii) m_x is a G_1-algebra morphism from u_x to v_x.

(iv) m_y is a G_0-algebra morphism from u_y to v_y.

(v) θ is natural.

It is natural to ask how the initial $\widehat{F}$-algebra might be characterized. Due to the fact that $\widehat{F}$-algebra morphisms consist of G_i-algebra morphisms, it is reasonable to assume that the initial $\widehat{F}$-algebra is related closely to the initial G_i-algebras. In fact, the $\widehat{F}$-algebra (ζ, r) from (6) is initial.

Corollary 4.8 *Let $G_1 \overset{\theta}{\Rightarrow} G_0$ be a natural transformation between two $\mathcal{C}$-endofunctors which admit initial algebras. Let $F\colon \mathbf{2} \times \mathcal{C} \to \mathcal{C}$ be the parameterized endofunctor given by $F(i, \mathbb{1}) = G_i$ and $F(!, \mathbb{1}) = \theta$, and let $\widehat{F}$ be the $\mathcal{C}^{\to}$-endofunctor generated by F. Let $G_i a_i \xrightarrow{r_i} a_i$ be the initial G_i-algebra, and let ζ be the G_1-algebra morphism given in (6). Then the initial $\widehat{F}$-algebra is (ζ, r).*

This result follows as an application of Theorem 4.5. It is the simplest case where the parameter category $\mathcal{B}$ is not discrete. The direct proof is given by Chuang and Lin and applied to give inductive semantics to dependent types [5].

5 Conclusions and future work

For a higher-order endofunctor $\widehat{F}$ that is generated from a parameterized endofunctor F, the existence of an initial $\widehat{F}$-algebra (resp. coalgebra) coincides exactly with F being initially (resp. finally) suitable. With the weakest of assumptions, this result generalizes and synthesizes several disparate observations made in the literature. It leads to the conclusion that effort should

be focused on systematically studying algebraic and coalgebraic properties of higher-order endofunctors. Future work includes identifying other useful instances of higher-order algebras and coalgebras.

Acknowledgement

The author gratefully acknowledges the helpful discussions with Jan-Li Lin who made him aware of the work on the arrow category [5]. Comments from the anonymous referees also improved this paper.

References

[1] Peter Aczel, Jirí Adámek, Stefan Milius, and Jiri Velebil. Infinite trees and completely iterative theories: a coalgebraic view. *Theor. Comput. Sci.*, 300(1-3):1–45, 2003.

[2] Robert Atkey. Parameterized notions of computation. In *MSFP 2006*, July 2006.

[3] Michael Barr and Charles Wells. *Toposes, Triples and Theories*. Springer-Verlag, 1985.

[4] Richard Blute, J. R. B. Cockett, and R. A. G. Seely. Categories for computation in context and unified logic. *Journal of Pure and Applied Algebra*, 116:49–98, 1997.

[5] Tyng-Ruey Chuang and Jan-Li Lin. An algebra of dependent data types. Technical Report TR-IIS-06-012, Institute of Information Science, Academia Sinica, 2006.

[6] Patricia Johann and Neil Ghani. Foundations for structured programming with gadts. In *POPL*, pages 297–308, 2008.

[7] Alexander Kurz and Dirk Pattison. Coalgebras and modal logic for parameterised endofunctors. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands, 2000.

[8] J. J. M. M. Rutten. Algebraic specification and coalgebraic synthesis of mealy machines. Technical report, In: Proceedings of FACS 2005. ENTCS, 2006.

[9] Tarmo Uustalu. Generalizing substitution. *ITA*, 37(4):315–336, 2003.

[10] Tarmo Uustalu and Varmo Vene. Comonadic notions of computation. *Electron. Notes Theor. Comput. Sci.*, 203(5):263–284, 2008.

Structural Operational Semantics and Modal Logic, Revisited

Bartek Klin[1]

University of Cambridge, Warsaw University

Abstract

A previously introduced combination of the bialgebraic approach to structural operational semantics with coalgebraic modal logic is re-examined and improved in some aspects. Firstly, a more abstract, conceptual proof of the main compositionality theorem is given, based on an understanding of modal logic as a study of coalgebras in slice categories of adjunctions. Secondly, a more concrete understanding of the assumptions of the theorem is provided, where proving compositionality amounts to finding a syntactic distributive law between two collections of predicate liftings.

Keywords: structural operational semantics, modal logic, coalgebra

1 Introduction

Compositionality of process equivalences is an important issue in the theory of Structural Operational Semantics (SOS; see e.g. [1,7]). Compositionality proofs for specific languages are often tedious, therefore plenty of meta-results have been proved that guarantee the compositionality of various equivalences by subjecting operational specifications to certain syntactic restrictions, called formats.

The process of inducing well-behaved transition systems from SOS specifications has been explained at the abstraction level of coalgebras, in the bialgebraic framework of [28]. There, a well-known SOS format called GSOS was understood as a type of distributive laws between behaviour and syntax endofunctors. The fact that LTS bisimilarity on GSOS-induced specifications is compositional, was explained at that level of generality.

[1] This work was supported by EPSRC grant EP/F042337/1. Email: klin@mimuw.edu.pl

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

One way to extend that approach to equivalences other than bisimilarities is to understand them as logical equivalences for some modal logics, and use a general coalgebraic approach to modal logic as developed, e.g., in [4,5,10,12,16,18,19,23,27]. In [15,17], such a combination of the bialgebraic approach with coalgebraic modal logic was presented. To prove that an equivalence defined by some logic on a transition system induced from an SOS specification is a congruence, one needs to equip logical formulas with a coalgebraic behaviour, and exhibit a "logical distributive law" of logical syntax over that behaviour that reflects the distributive law modeling the SOS specification.

Looking from some perspective, neither mathematical economy nor practical usability of [15,17] was entirely satisfactory. Firstly, the proof of the main compositionality result, albeit elementary, involved plenty of diagram chasing and inductive proofs, and in general was not very illuminating. Second, perhaps more painful deficiency, was that no intuitive general understanding of coalgebraic behaviour for logical formulas was provided. Although logical distributive laws for some specific kinds of logical behaviour were presented in an appealing, SOS-like manner, no concrete understanding of such laws for other types of behaviour was found. Also, no guidelines to finding behaviour functors for logical formulas were given, other than wild guessing. Checking that a candidate logical distributive law was correct involved heavy calculations of complex natural transformations, far removed from common understanding of formulas and processes.

This paper is an attempt to remove these two deficiencies to some degree. First, a more abstract, conceptual proof of the main compositionality theorem of [15] is provided (and the theorem is mildly generalized in the process). To this end, the interpretation of modal logic in coalgebras is understood as a functor from the category of coalgebras to a slice category of an adjunction. The compositionality theorem then follows from lifting that functor to structures that involve process syntax, via an adjoint lifting theorem.

Secondly, a concrete understanding of the compositionality theorem is provided for the important example where both processes and formulas live in the category of sets. Coalgebraic behaviour for formulas is explained in terms of predicate liftings [22,27], and logical distributive laws are syntactic distributive laws between two collections of liftings. Proving compositionality of a logical equivalence amounts to finding a suitable collection of liftings for process syntax, together with a set of equations that involve those liftings. Since the notion of predicate lifting is well studied and understood, this formulation should hopefully aid the understanding of our bialgebraic approach to logical compositionality.

The paper is structured as follows. In Sections 2 and 3, the bialgebraic approach to SOS and coalgebraic modal logic are briefly recalled. Section 4

studies endofunctors, algebras and coalgebras in slice categories of adjunctions, and culminates in a proof of the main compositionality theorem. Section 5 provides a concrete interpretation of the theorem in terms of predicate liftings. Finally, Sections 6 explains how both deficiencies mentioned above persist to some degree in the present formulation. Some proofs, not essential for the main line of reasoning, are relegated to Appendix.

Parts of the paper might be of interest also to those readers who do not care much about compositionality or SOS. For an explanation of coalgebraic modal logic in terms of (co)algebras in slice categories of adjunctions, without any involvement of process syntax, it is enough to read Sections 3, 4.1 and 4.2.

The reader is expected to be acquainted with basic category theory ([21] is a standard reference) and with the coalgebraic approach to theory of systems [26].

Acknowledgment. The author is grateful to Ichiro Hasuo for many interesting discussions.

2 SOS and distributive laws

In the context of Structural Operational Semantics, transition systems of various kinds are defined by structural induction using inference rules, and have closed terms over some signature as states. For example, given a fixed set A of labels, the set of rules

$$\frac{}{a \xrightarrow{a} \mathsf{0}} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{a} y'}{x \otimes y \xrightarrow{a} x' \otimes y'} \tag{1}$$

(where a ranges over A) inductively defines a labeled transition system (LTS) on the set of closed terms over the grammar:

$$t ::= \mathsf{0} \mid a \mid t \otimes t \qquad\qquad (a \in A). \tag{2}$$

In [28], this situation was expressed in the coalgebraic setting with the use of distributive laws and bialgebras for them. For example, rules (1) define a natural transformation $\lambda : \Sigma B \Longrightarrow B\Sigma$ (see e.g. [17] for a gentle explanation of this construction), where Σ is the polynomial endofunctor on **Set** corresponding to the grammar (2), and $B = (\mathcal{P}_\omega -)^A$, where $\mathcal{P}_\omega$ is the finite powerset endofunctor; B-coalgebras are image-finite A-labeled transition systems (LTSs).

For any endofunctors Σ and B on a category $\mathcal{C}$, a transformation as above, called a distributive law of Σ over B, induces an endofunctor Σ_λ on the category B**-coalg** of B-coalgebras, and an endofunctor B_λ on the category Σ**-alg**

of Σ-algebras, acting on objects as follows:

$$\begin{aligned}\Sigma_\lambda(\, X \xrightarrow{h} BX\,) &= \Sigma X \xrightarrow{\Sigma h} \Sigma BX \xrightarrow{\lambda_X} B\Sigma X\\ B_\lambda(\, \Sigma X \xrightarrow{g} X\,) &= \Sigma BX \xrightarrow{\lambda_X} B\Sigma X \xrightarrow{Bg} BX\end{aligned}$$

and as Σ (resp. B) on morphisms. Clearly Σ_λ lifts Σ and B_λ lifts B along the respective forgetful functors $U_B : B\textbf{-coalg} \to \mathcal{C}$ and $U^\Sigma : \Sigma\textbf{-alg} \to \mathcal{C}$.

It is easy to see that a Σ_λ-algebra, or a B_λ-coalgebra, consists of a Σ-algebra g and a B-coalgebra h with the same carrier, so that the diagram:

$$\begin{array}{ccccc}\Sigma X & \xrightarrow{g} & X & \xrightarrow{h} & BX\\ {\scriptstyle \Sigma h}\downarrow & & & & \uparrow{\scriptstyle Bg}\\ \Sigma BX & & \xrightarrow[\lambda_X]{} & & B\Sigma X\end{array} \tag{3}$$

commutes. Such structures are called λ-bialgebras (with carrier X), and a λ-bialgebra morphism is a map in $\mathcal{C}$ between the respective carriers that is simultaneously an algebra morphism and a coalgebra morphism; this defines a category $\lambda\textbf{-bialg}$ of λ-bialgebras. There is an isomorphism of categories and a commuting square of forgetful functors:

$$\lambda\textbf{-bialg} \cong \Sigma_\lambda\textbf{-alg} \cong B_\lambda\textbf{-coalg} \qquad \begin{array}{ccc}\lambda\textbf{-bialg} & \xrightarrow{U^{\Sigma_\lambda}} & B\textbf{-coalg}\\ {\scriptstyle U_{B_\lambda}}\downarrow & & \downarrow{\scriptstyle U_B}\\ \Sigma\textbf{-alg} & \xrightarrow[U^\Sigma]{} & \mathcal{C}.\end{array} \tag{4}$$

If g is an initial Σ-algebra, then there is a unique B-coalgebra h such that (3) commutes, defined as the unique Σ-algebra morphism from g to $B_\lambda(g)$. The result is an initial λ-bialgebra, and if λ corresponds to an SOS specification as in (1), then h corresponds to the transition system induced by the specification.

Dually, if a final B-coalgebra exists, it extends uniquely to a final λ-bialgebra. This immediately implies:

Proposition 2.1 *For any λ-bialgebra $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$, the unique coalgebra morphism from h to the final B-coalgebra is a Σ-algebra morphism from g.*

When $\mathcal{C} = \textbf{Set}$, two elements $x, y \in X$ for a given coalgebra $h : X \to BX$ are called *observationally equivalent* if they are identified by some coalgebra morphism. In particular, if final B-coalgebras exist, observational equivalence on h is the kernel relation of the final coalgebra morphism from h. Thus Proposition 2.1, applied to initial bialgebras, means that observational equivalence on the transition system induced by a specification is a congruence, if the

specification corresponds to a distributive law λ.[2]

The practical applicability of Proposition 2.1 as stated here is rather limited, since few interesting examples actually correspond to λ as above. Already in [28] more general laws were studied, involving the free pointed endofunctor $\mathrm{Id} + \Sigma$ and the free monad Σ^*, and the cofree copointed endofunctor $\mathrm{Id} \times B$ and cofree comonad B^ω (assuming they exist). In particular, distributive laws of Σ^* over $\mathrm{Id} \times B$ correspond bijectively [20] to natural transformations $\lambda : \Sigma(\mathrm{Id} \times B) \Longrightarrow B\Sigma^*$, and for $B = (\mathcal{P}_\omega -)^A$, these correspond [2] to SOS specifications in the well-studied GSOS format [3]. Similarly, distributive laws of $\mathrm{Id} + \Sigma$ over B^ω correspond to a format called safe-ntree in [28]. Proposition 2.1 is proved without much change for each of these more expressive laws; in fact, one does not need to prove each case separately, as each type of laws in question induces distributive laws of the monad Σ^* over the comonad B^ω along the lines of [20], and Proposition 2.1 works for such laws as well, with essentially the same proof.

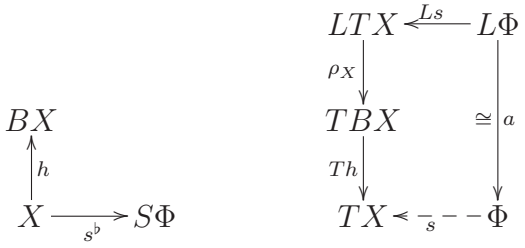
In this paper, only simple distributive laws $\lambda : \Sigma B \Longrightarrow B\Sigma$ are considered. This is mainly to simplify the presentation and save space in the technical development in Section 4. The general case of distributive laws of monads over comonads is dealt with in an entirely analogous manner, but with additional checks to ensure the existence of certain (co)free (co)monads and the compatibility of (co)units and (co)multiplications, and is better left to an extended version of this paper.

3 Coalgebraic modal logic

An abstract approach to modal logics for coalgebras, based on adjunctions of contravariant functors, has attracted considerable attention (e.g., [4,5,10,12,16,18,27]). Assume an adjunction $S^{\mathrm{op}} \dashv T : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}$, with the intuition that objects of $\mathcal{D}$ are sets (or structures) of formulas, and objects of $\mathcal{C}$ are sets (or structures) of processes or states. For an endofunctor B on $\mathcal{C}$, a coalgebraic modal logic for B-coalgebras is given by an endofunctor L on $\mathcal{D}$ (the logical syntax), and a natural transformation $\rho : LT \Longrightarrow TB^{\mathrm{op}}$ (called a *connection*). Under the assumption that an initial L-algebra $a : L\Phi \to \Phi$ exists (intuitively, it is an algebra of logical formulas), the interpretation of logic (L, ρ) on a given coalgebra $h : X \to BX$ is obtained by transposing along the adjunction $S^{\mathrm{op}} \dashv T$

[2] In [28], the congruence result is proved for coalgebraic bisimilarity [26] rather than observational equivalence; for that, the additional assumption of B preserving weak pullbacks was needed.

the unique L-algebra morphism s from a to $Th \circ \rho_X$, as in the diagram:



(5)

where the left part is drawn in $\mathcal{C}$ and the right part in $\mathcal{D}$.

Take for example $\mathcal{C} = \mathcal{D} = \mathbf{Set}$ and $S = T = 2^-$ (where $2 = \{\mathtt{tt}, \mathtt{ff}\}$), and $B = (\mathcal{P}_\omega -)^A$. The trace fragment of Hennessy-Milner logic for B-coalgebras (i.e. LTSs) has syntax described by the grammar $\phi ::= \top \mid \langle a \rangle \phi$ that corresponds to the endofunctor $L\Phi = 1 + A \times \Phi$, and its standard semantics corresponds to $\rho_X : L2^X \to 2^{BX}$ defined by:

$$\begin{aligned} \rho_X(\top)(b) &= \mathtt{tt} \text{ always} \\ \rho_X(\langle a \rangle \phi)(b) = \mathtt{tt} &\iff \exists y \in b(a).\ \phi(y) = \mathtt{tt} \end{aligned} \tag{6}$$

for any X. It is straightforward to check that (the kernel relation of) the interpretation of this (L, ρ) on a B-coalgebra is the trace equivalence on it.

When searching for a logic for a given B, one may often restrict attention to endofunctors of a certain shape, without losing any generality. We now briefly recall an analysis from [16]. Connections $\rho : LT \Longrightarrow TB^{\mathrm{op}}$ are in bijective correspondence with natural transformations $\tilde{\rho} : L \Longrightarrow TB^{\mathrm{op}}S^{\mathrm{op}}$ by $\tilde{\rho} = \rho S^{\mathrm{op}} \circ L\eta$, where $\eta : \mathrm{Id}_{\mathcal{D}} \to TS^{\mathrm{op}}$ is the unit of $S^{\mathrm{op}} \dashv T$. If one insists on L being finitary, then $\tilde{\rho}$ factors through the finitary restriction of $TB^{\mathrm{op}}S^{\mathrm{op}}$. The latter is defined by a coend formula; without losing generality one may replace the coend with a coproduct and, for $\mathcal{D} = \mathbf{Set}$, require a natural transformation

$$\tilde{\rho} : L \Longrightarrow \coprod_{n \in \mathbb{N}} TBSn \times (-)^n.$$

For $S = T = 2^-$ and $n \in \mathbb{N}$, elements of $TBSn$ are functions $\beta : B(2^n) \to 2$, which we call n-ary B-*modalities*. (By Yoneda Lemma, these bijectively correspond to natural transformations $\beta^{\mathcal{Y}} : (2^-)^n \Longrightarrow 2^{B-}$, i.e., polyadic predicate liftings of [27].) As a result, a finitary L with ρ can be presented as a collection of B-modalities: a family $(L_n)_{n \in \mathbb{N}}$ of sets $L_n \subseteq 2^{B2^n}$ represents the polynomial endofunctor

$$L = \coprod_{n \in \mathbb{N}} L_n \times (-)^n, \tag{7}$$

with $\rho : L(2^-) \Longrightarrow 2^{B-}$ defined by copairing all predicate liftings $\beta^{\mathcal{Y}} : (2^-)^n \Longrightarrow 2^{B-}$ for each $n \in \mathbb{N}$ and $\beta \in L_n$.

For example, the trace logic for $B = (\mathcal{P}_\omega -)^A$ is represented by the following collection of modalities: $L_0 = \{\top\}$, $L_1 = \{\langle a\rangle \mid a \in A\}$, $L_n = \emptyset$ for $n > 1$, where $\top : B1 \to 2$ and $\langle a\rangle : B2 \to 2$ are defined by:

$$\top(b) = \mathtt{tt} \text{ always}, \qquad \langle a\rangle(b) = \mathtt{tt} \iff \mathtt{tt} \in b(a). \tag{8}$$

We mention in passing that this approach to logics suffers from practical expressivity problems similar to those mentioned in Section 2. For example, although a version of finitary Hennessy-Milner logic [8] for bisimilarity can be defined this way, it is rather unwieldy, with infinitely many modalities of arbitrary arities (see [16]). This is because logics based on $S = T = 2^-$ lack in-built support for propositional connectives, which must then be encoded as parts of complex modalities.

One way to avoid this problem is to change the adjunction $S^{\mathrm{op}} \dashv T$ in question (see [12] for examples). Another way is to consider, by analogy to distributive laws and SOS, more general types of connections. For example, one can allow ones like $\rho : LT \Longrightarrow T(\mathrm{Id} \times B)^{op}$, whereby propositional connectives such as $\wedge$ can easily be defined as simple modalities. One can even consider connections such as $\rho : LT \Longrightarrow T(B^\omega)^{\mathrm{op}}$ to describe e.g. Hennessy-Milner logic for weak bisimilarity.

Again, in this paper only simple connections $\rho : LT \Longrightarrow TB$ are considered; the issue of coalgebraic modal logics based on more complex connections is left for a separate study.

4 Compositionality for logical equivalences

Our main technical goal is to modify Proposition 2.1 to deal with logical equivalences rather than with observational equivalence. We shall prove (in Theorem 4.6) that under certain assumptions, for any λ-bialgebra $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$, the interpretation of logic (L, ρ) on h is a Σ-algebra morphism from g. To formulate the theorem and its proof, we introduce some basic notions and results regarding (co)algebras on slice categories of adjunctions. To structure the development to some degree, we shall begin with slice categories of functors, and later see what additional structure the adjunction $S^{\mathrm{op}} \dashv T$ introduces.

4.1 Slice categories

For a functor $T : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}$, the *slice category* $(\mathcal{D}{\downarrow}T)$ has:

- as objects, triples $\langle \Phi, X, s\rangle$ with $\Phi \in \mathcal{D}$, $X \in \mathcal{C}$ and $s : \Phi \to TX$ in $\mathcal{D}$,
- as maps from $\langle \Phi, X, s\rangle$ to $\langle \Psi, Y, r\rangle$, pairs (g, f) with $g : \Phi \to \Psi$ in $\mathcal{D}$ and

$f : Y \to X$ in $\mathcal{C}$ such that

$$\begin{array}{ccc} \Phi & \xrightarrow{s} & TX \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle Tf} \\ \Psi & \xrightarrow[r]{} & TY \end{array} \tag{9}$$

commutes.

There are obvious projection functors, denoted $\Pi_1 : (\mathcal{D}\downarrow T) \to \mathcal{D}$ and $\Pi_2 : (\mathcal{D}\downarrow T) \to \mathcal{C}^{\mathrm{op}}$.

If an initial object 0 in $\mathcal{D}$ exists, it determines a full embedding $0^{\to} : \mathcal{C}^{\mathrm{op}} \to (\mathcal{D}\downarrow T)$ defined on objects by $0^{\to}(X) = \langle 0, X, 0 : 0 \to TX\rangle$, where the arrow 0 is unique by initiality, and on arrows by $0^{\to}(g) = \langle \mathrm{id}_0, g\rangle$. It is easy to verify that:

Proposition 4.1 *$0^{\to}$ is left adjoint to $\Pi_2 : (\mathcal{D}\downarrow T) \to \mathcal{C}^{\mathrm{op}}$, and the unit of the adjunction is the identity natural transformation.* □

Sliced endofunctors. Assume endofunctors $B : \mathcal{C} \to \mathcal{C}$ and $L : \mathcal{D} \to \mathcal{D}$, and a natural transformation $\rho : LT \Longrightarrow TB^{\mathrm{op}}$.

These ingredients define an endofunctor on $(\mathcal{D}\downarrow T)$, denoted $\widehat{\rho}$, as follows:

- on objects, $\widehat{\rho}\,\langle \Phi, X, r\rangle = \langle L\Phi, BX, \rho_X \circ Lr\rangle$
- on maps, $\widehat{\rho}\,\langle g, f\rangle = \langle Lg, Bf\rangle$.

It is easy to check that this is well-defined and functorial. Clearly $\widehat{\rho}$ lifts L along Π_1, in the sense that $\Pi_1 \circ \widehat{\rho} = L \circ \Pi_1$. Similarly, $\widehat{\rho}$ lifts B^{op} along Π_2.

Endofunctors on $(\mathcal{D}\downarrow T)$ that arise in this way will be called *sliced (by ρ)*. Not every endofunctor on $(\mathcal{D}\downarrow T)$ is sliced in general, even if T is well-behaved (for a counterexample, see the Appendix). However:

Proposition 4.2 *Consider an endofunctor $K : (\mathcal{D}\downarrow T) \to (\mathcal{D}\downarrow T)$ such that for some $L : \mathcal{D} \to \mathcal{D}$ and $B : \mathcal{C} \to \mathcal{C}$, K lifts L along Π_1 and B^{op} along Π_2. Then K is sliced in a unique way.*

Proof. See the Appendix. □

It immediately follows that sliced endofunctors are closed under composition. However, a more direct proof is possible: for $\rho : LT \Longrightarrow TB^{\mathrm{op}}$ and $\rho' : L'T \Longrightarrow T(B')^{\mathrm{op}}$, it is easy to check that the composite endofunctor $\widehat{\rho}\widehat{\rho'}$ is sliced by:

$$\rho(B')^{\mathrm{op}} \circ L\rho' : LL'T \Longrightarrow T(BB')^{\mathrm{op}}. \tag{10}$$

Sliced natural transformations. Assume connections $\rho : LT \Longrightarrow TB^{\mathrm{op}}$ and $\rho' : L'T \Longrightarrow TB'^{\mathrm{op}}$. Any two natural transformations $\alpha : L \Longrightarrow L'$ and

$\beta : B' \Longrightarrow B$ such that

$$\begin{array}{ccc} LTS & \xRightarrow{\rho} & TB^{op} \\ {\scriptstyle \alpha T}\Downarrow & & \Downarrow{\scriptstyle T\beta^{op}} \\ L'T & \xRightarrow[\rho']{} & TB'^{op} \end{array} \tag{11}$$

commutes, give rise to a natural transformation $\alpha \circledast \beta : \widehat{\rho} \Longrightarrow \widehat{\rho'}$ defined by:

$$\alpha \circledast \beta_{\langle \Phi, X, s\rangle} = \langle \alpha_\Phi, \beta_X \rangle \,. \tag{12}$$

Not every transformation between sliced endofunctors is of this form (for a counterexample, see the Appendix). However, in Section 4.2 we shall show that this is the case if T has a left adjoint.

Algebras. Given a connection $\rho : LT \Longrightarrow TB^{\mathrm{op}}$, a $\widehat{\rho}$-algebra is, equivalently, an L-algebra $g : L\Phi \to \Phi$ in $\mathcal{D}$, a B-coalgebra $h : X \to BX$ in $\mathcal{C}$, and a map $s : \Phi \to TX$, such that the diagram

$$\begin{array}{ccccc} L\Phi & \xrightarrow{Ls} & LTX & \xrightarrow{\rho_X} & TBX \\ {\scriptstyle g}\downarrow & & & & \downarrow{\scriptstyle Th} \\ \Phi & & \xrightarrow[s]{} & & TX \end{array} \tag{13}$$

commutes in $\mathcal{D}$. Moreover, $\widehat{\rho}$-algebra morphisms are easily seen to be pairs of an L-algebra morphism and a B-coalgebra morphisms. In particular, there are evident projection functors $\Pi_1 : \widehat{\rho}\textbf{-alg} \to L\textbf{-alg}$ and $\Pi_2 : \widehat{\rho}\textbf{-alg} \to (B\textbf{-coalg})^{\mathrm{op}}$.

Let us pause for a moment to reflect on the meaning of $\widehat{\rho}$-algebras: they are B-coalgebras h (systems) together with L-algebras g (logical theories) interpreted in them (via s). For example, if $\mathcal{C} = \mathcal{D} = \mathbf{Set}$ and $S = T = 2^-$, the function $s : \Phi \to TX$ in the carrier of a $\widehat{\rho}$-algebra is just a relation between Φ and X. For $B = (\mathcal{P}_\omega -)^A$ and (L, ρ) as in (6), the equivalence relation on X defined by this relation is always contained in trace equivalence on h, and coincides with it if g is initial. Morphisms of $\widehat{\rho}$-algebras reflect these equivalence relations, implicitly present in their carriers. This suggests that when one wants to study coalgebras "up to" some logical equivalence, and when the task of finding an explicit coalgebraic presentation of these "up to" structures (such as in [11]) seems difficult or simply not worthwhile, one may try to resort to implicit modeling of logical equivalences by theories interpreted in coalgebras; this view is advocated e.g. in [23], and the present paper may be considered as an example application of it. One may argue that, just as structural operational semantics is a study of coalgebra in categories of algebras, coalgebraic modal logic is a study of (co)algebra in slice categories.

Back to the formal development: an alternative reading of (13) is that s is a L-algebra morphism. In other words, $\widehat{\rho}$-algebras are morphisms be-

tween L-algebras of a certain shape. To formalize this, observe that ρ induces a functor $\overline{T} : (B\textbf{-coalg})^{\mathrm{op}} \to L\textbf{-alg}$ defined by $\overline{T}(X \xrightarrow{h} BX) = LTX \xrightarrow{\rho_X} TBX \xrightarrow{Th} TX$ on B-coalgebras, and as T on B-coalgebra morphisms.

Proposition 4.3 $\widehat{\rho}\textbf{-}\boldsymbol{alg} \cong (L\textbf{-}\boldsymbol{alg}{\downarrow}\overline{T})$.

Proof. We have already essentially noticed the correspondence on objects; morphisms are equally easy. □

Corollary 4.4 *If an initial L-algebra exists, then the projection functor $\Pi_2 : \widehat{\rho}\textbf{-}\boldsymbol{alg} \to (B\textbf{-}\boldsymbol{coalg})^{\mathrm{op}}$ has a left adjoint, and the unit of the adjunction is the identity natural transformation.*

Proof. Use Proposition 4.3 and apply Proposition 4.1. Given an initial L-algebra a, the left adjoint will be denoted $a^{\to}$. □

Note that the composition of $a^{\to}$ with the forgetful functor from $\widehat{\rho}$**-alg**:

$$(B\textbf{-coalg})^{\mathrm{op}} \underset{\Pi_2}{\overset{a^{\to}}{\underset{\longleftarrow}{\longrightarrow}}} \widehat{\rho}\textbf{-alg} \xrightarrow{U^{\widehat{\rho}}} (\mathcal{D}{\downarrow}T)$$

corresponds almost entirely to the interpretation of coalgebraic modal logic in B-coalgebras, as constructed in (5). Indeed, the only step missing in this functorial presentation is the transposition of the semantic map s from the initial L-algebra. For this final step, obviously, it is crucial that the functor T has a left adjoint; we shall now proceed to develop our theory further with this additional assumption.

4.2 Slice categories of adjunctions

In this section, we shall assume that $T : \mathcal{C}^{\mathrm{op}} \to \mathcal{D}$ has a left adjoint $S^{\mathrm{op}} : \mathcal{D} \to \mathcal{C}^{\mathrm{op}}$. The unit and counit of the adjunction $S^{\mathrm{op}} \dashv T$ will be denoted $\eta : \mathrm{Id} \Longrightarrow TS^{\mathrm{op}}$ and $\epsilon : S^{\mathrm{op}}T \Longrightarrow \mathrm{Id}$ respectively. Obviously then $T^{\mathrm{op}} : \mathcal{C} \to \mathcal{D}^{\mathrm{op}}$ is left adjoint to $S : \mathcal{D}^{\mathrm{op}} \to \mathcal{C}$, and $\eta^{\mathrm{op}} : T^{\mathrm{op}}S \Longrightarrow \mathrm{Id}$ and $\epsilon^{\mathrm{op}} : \mathrm{Id} \Longrightarrow ST^{\mathrm{op}}$ are the counit and the unit of the adjunction $T^{\mathrm{op}} \dashv S$. This adjoint situation means that there is a bijection

$$\mathcal{C}(X, S\Phi) \cong \mathcal{D}(\Phi, TX)$$

natural in $X \in \mathcal{C}$ and $\Phi \in \mathcal{D}$; we shall abuse notation and denote both sides of this bijection by $-^{\flat}$. A defining property of adjunctions is the isomorphism of slice categories: $(\mathcal{D}{\downarrow}T) \cong (\mathcal{C}{\downarrow}S)^{\mathrm{op}}$ (the isomorphism maps an object $\langle \Phi, X, s \rangle$ to $\langle X, \Phi, s^{\flat} \rangle$).

Coalgebraic modal logic as a functor. One immediate consequence of the adjunction assumption is that one can represent the entire modal logic

interpretation construction (5) as a functor from the category of B-coalgebras:

$$(B\textbf{-coalg})^{\mathrm{op}} \underset{\Pi_2}{\overset{g^{\rightarrow}}{\underset{\perp}{\rightleftarrows}}} \widehat{\rho}\textbf{-alg} \xrightarrow{U^{\widehat{\rho}}} (\mathcal{D}{\downarrow}T) \cong (\mathcal{C}{\downarrow}S)^{\mathrm{op}} \tag{14}$$

However, the most useful consequences of that assumption appear when one decides to study *co*algebras for sliced endofunctors on $(\mathcal{D}{\downarrow}T)$.

Coalgebras. In the situation considered in Section 4.1, categories of coalgebras for sliced endofunctors have, in general, considerably less structure than those of algebras.

Consider endofunctors Σ on $\mathcal{C}$ and Γ on $\mathcal{D}$, and a connection $\zeta : \Gamma T \Longrightarrow T\Sigma^{\mathrm{op}}$ as in Section 4.1. A $\widehat{\zeta}$-coalgebra is a Γ-coalgebra $h : \Phi \to \Gamma\Phi$ in $\mathcal{D}$, a Σ-algebra $g : \Sigma X \to X$ in $\mathcal{C}$, and a map $s : \Phi \to TX$, such that the diagram

$$\begin{array}{ccccc} \Gamma\Phi & \xrightarrow{\Gamma s} & \Gamma TX & \xrightarrow{\zeta_X} & T\Sigma X \\ {\scriptstyle h}\uparrow & & & & \uparrow{\scriptstyle Tg} \\ \Phi & & \xrightarrow[s]{} & & TX \end{array} \tag{15}$$

commutes in $\mathcal{D}$, and a $\widehat{\zeta}$-coalgebra morphisms is a pair of a Γ-coalgebra morphism and a Σ-algebra morphism. This gives projection functors $\Pi_1 : \widehat{\zeta}\textbf{-coalg} \to \Gamma\textbf{-coalg}$ and $\Pi_2 : \widehat{\zeta}\textbf{-coalg} \to (\Sigma\textbf{-alg})^{\mathrm{op}}$.

In general, contrary to the situation of sliced algebras, the diagram (15) cannot be read as a coalgebra morphism (in [23], it was called a "twisted coalgebra morphism"). As a result, no property analogous to Proposition 4.3 holds for $\widehat{\zeta}$-coalgebras in general. However, additional structure appears when we assume a left adjoint $S^{\mathrm{op}} \dashv T$. Indeed, then connections $\zeta : \Gamma T \Longrightarrow T\Sigma^{\mathrm{op}}$ are in bijective correspondence with their *adjoint mates* [14] $\zeta^{\star} : \Sigma S \Longrightarrow S\Gamma^{\mathrm{op}}$, defined by transposing $\zeta S^{\mathrm{op}} \circ \Gamma\eta : \Gamma \Longrightarrow T\Sigma^{\mathrm{op}} S^{\mathrm{op}}$. It is straightforward to check that $\widehat{\zeta^{\star}}$ coincides with $(\widehat{\zeta})^{\mathrm{op}}$ along the isomorphism $(\mathcal{C}{\downarrow}S) \cong (\mathcal{D}{\downarrow}T)^{\mathrm{op}}$. In particular, this implies an isomorphism

$$\widehat{\zeta}\textbf{-coalg} \cong (\widehat{\zeta^{\star}}\textbf{-alg})^{\mathrm{op}}. \tag{16}$$

Natural transformations are sliced. Finally, the fact that T has a left adjoint implies that all natural transformations between sliced endofunctors are sliced. Indeed, consider any $\rho : \Gamma T \Longrightarrow T\Sigma^{\mathrm{op}}$ and $\rho' : \Gamma' T \Longrightarrow T\Sigma'^{\mathrm{op}}$.

Proposition 4.5 *If $S^{\mathrm{op}} \dashv T$ then natural transformations $\kappa : \widehat{\rho} \Longrightarrow \widehat{\rho'}$ are in one-to-one correspondence with pairs $\langle \alpha : \Gamma \Longrightarrow \Gamma', \beta : \Sigma' \Longrightarrow \Sigma \rangle$ such that $T\beta^{\mathrm{op}} \circ \rho = \rho' \circ \alpha T$ as in (11).*

Proof. (12) shows how to define κ from α and β. For the other direction, see the Appendix. □

4.3 Sliced distributive laws and compositionality

We now proceed to the study of bialgebras for distributive laws between sliced endofunctors.

Sliced distributive laws. For an adjunction $S^{\mathrm{op}} \dashv T : \mathcal{C}^{\mathrm{op}} \to D$, consider endofunctors B, Σ on $\mathcal{C}$ and L, Γ on $\mathcal{D}$, together with connections $\rho : LT \to TB^{\mathrm{op}}$ and $\zeta : \Gamma T \Longrightarrow T\Sigma^{\mathrm{op}}$ that define sliced endofunctors $\widehat{\rho}, \widehat{\zeta}$ on $(\mathcal{D}{\downarrow}T)$.

Now assume a distributive law of $\widehat{\rho}$ over $\widehat{\zeta}$, i.e., a natural transformation $\kappa : \widehat{\rho}\widehat{\zeta} \Longrightarrow \widehat{\zeta}\widehat{\rho}$. By (10), both $\widehat{\rho}\widehat{\zeta}$ and $\widehat{\zeta}\widehat{\rho}$ are sliced, and further by Proposition 4.5, κ is of the form $\kappa = \chi \circledast \lambda$ (see (12)) for some distributive laws $\chi : L\Gamma \Longrightarrow \Gamma L$ and $\lambda : \Sigma B \Longrightarrow B\Sigma$ such that the hexagon of natural transformations commutes (cf. (11)):

$$\begin{array}{ccccc} L\Gamma T & \xRightarrow{L\zeta} & LT\Sigma^{\mathrm{op}} & \xRightarrow{\rho\Sigma^{\mathrm{op}}} & T(B\Sigma)^{\mathrm{op}} \\ {\scriptstyle \chi T}\Downarrow & & & & \Downarrow{\scriptstyle T\lambda^{\mathrm{op}}} \\ \Gamma LT & \xRightarrow[\Gamma\rho]{} & \Gamma TB^{\mathrm{op}} & \xRightarrow[\zeta B^{\mathrm{op}}]{} & T(\Sigma B)^{\mathrm{op}}. \end{array} \tag{17}$$

Sliced bialgebras. As in Section 2, the law κ defines endofunctors $\widehat{\rho}_\kappa$ on $\widehat{\zeta}$**-coalg** and $\widehat{\zeta}_\kappa$ on $\widehat{\rho}$**-alg**, with an isomorphism of categories and a commuting square of forgetful functors (cf. (4)):

$$\kappa\textbf{-bialg} \cong \widehat{\rho}_\kappa\textbf{-alg} \cong \widehat{\zeta}_\kappa\textbf{-coalg} \qquad \begin{array}{ccc} \kappa\textbf{-bialg} & \xrightarrow{U^{\widehat{\rho}_\kappa}} & \widehat{\zeta}\textbf{-coalg} \\ {\scriptstyle U_{\widehat{\zeta}_\kappa}}\downarrow & & \downarrow{\scriptstyle U_{\widehat{\zeta}}} \\ \widehat{\rho}\textbf{-alg} & \xrightarrow[U^{\widehat{\rho}}]{} & (\mathcal{D}{\downarrow}T). \end{array} \tag{18}$$

To convey some intuition, it might be useful to provide a more concrete descriptions of κ-bialgebras. Each of these consists of a χ-bialgebra $L\Phi \xrightarrow{k} \Phi \xrightarrow{l} \Gamma\Phi$, a λ-bialgebra $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$ and an arrow $s : \Phi \to TX$ in $\mathcal{D}$, such that the diagram:

$$\begin{array}{ccccccc} L\Phi & & \xrightarrow{\quad k \quad} & & \Phi & & \xrightarrow{\quad l \quad} & & \Gamma\Phi \\ {\scriptstyle Ls}\downarrow & & & & \downarrow{\scriptstyle s} & & & & \downarrow{\scriptstyle \Gamma s} \\ LTX & \xrightarrow[\rho_X]{} & TBX & \xrightarrow[Th]{} & TX & \xrightarrow[Tg]{} & T\Sigma X & \xleftarrow[\zeta_X]{} & \Gamma TX \end{array} \tag{19}$$

commutes. Morphisms of κ-bialgebras are pairs of a χ- and a λ-bialgebra morphisms; in particular, there is an evident projection functor, which we will denote $\overline{\Pi_2} : \kappa\textbf{-bialg} \to (\lambda\textbf{-bialg})^{\mathrm{op}}$.

Lifting coalgebraic modal logic. Our immediate goal now is to exhibit a left adjoint to $\overline{\Pi_2}$. Note that the bottom row of (19) is not a χ-bialgebra, so

κ-**bialg** is not easily a slice category and the simple tactic of using Proposition 4.1 cannot be used. Instead, adjoint lifting can be used in the following way.

Since κ acts as λ on $\mathcal{C}$-components (see (12)), it is straightforward to check that $\widehat{\zeta}_\kappa$ acts as Σ_λ on the B-coalgebra components of $\widehat{\rho}$-algebras and their morphisms; formally, $\Pi_2 \circ \widehat{\zeta}_\kappa = (\Sigma_\lambda)^{\mathrm{op}} \circ \Pi_2 : \widehat{\rho}\textbf{-alg} \to (B\textbf{-coalg})^{\mathrm{op}}$. This defines a lifting of Π_2 to a functor from $\widehat{\zeta}_\kappa$-**coalg** to $(\Sigma_\lambda\textbf{-alg})^{\mathrm{op}}$ as in (A.1); it is straightforward to check that this lifted functor coincides with $\overline{\Pi_2}$, which justifies its name. We can now apply Proposition A.1 to

$$\begin{array}{ccc} (\lambda\textbf{-bialg})^{\mathrm{op}} & \xleftarrow{\overline{\Pi_2}} & \kappa\textbf{-bialg} \\ {\scriptstyle (U^{\Sigma_\lambda})^{\mathrm{op}}}\downarrow & & \downarrow{\scriptstyle U_{\widehat{\zeta}_\kappa}} \\ (B\textbf{-coalg})^{\mathrm{op}} & \xleftarrow[\Pi_2]{} & \widehat{\rho}\textbf{-alg} \end{array}$$

and obtain a left adjoint $\overline{a^{\rightarrow}} \dashv \overline{\Pi_2}$. Combined with (18) and (16), this completes a lifting of the coalgebraic modal logic semantics (14) as in the diagram:

$$\begin{array}{ccccccc} (\lambda\textbf{-bialg})^{\mathrm{op}} & \underset{\overline{\Pi_2}}{\overset{\overline{a^{\rightarrow}}}{\perp}} & \kappa\textbf{-bialg} & \xrightarrow{U^{\widehat{\rho}_\kappa}} & \widehat{\zeta}\textbf{-coalg} & \cong & (\widehat{\zeta^\star}\textbf{-alg})^{\mathrm{op}} \\ {\scriptstyle (U^{\Sigma_\lambda})^{\mathrm{op}}}\downarrow & & \downarrow{\scriptstyle U_{\widehat{\zeta}_\kappa}} & & \downarrow{\scriptstyle U_{\widehat{\zeta}}} & & \downarrow{\scriptstyle (U^{\widehat{\zeta^\star}})^{\mathrm{op}}} \\ (B\textbf{-coalg})^{\mathrm{op}} & \underset{\overline{\Pi_2}}{\overset{a^{\rightarrow}}{\perp}} & \widehat{\rho}\textbf{-alg} & \xrightarrow[U^{\widehat{\rho}}]{} & (\mathcal{D}\downarrow T) & \cong & (\mathcal{C}\downarrow S)^{\mathrm{op}} \end{array} \tag{20}$$

Note that, by Corollary 4.4 and by the remark after Proposition A.1, the monad $\overline{\Pi_2} \circ \overline{a^{\rightarrow}}$ is (naturally isomorphic to) identity. This, together with the evident commuting square of forgetful and projection functors:

$$\begin{array}{ccccc} (\lambda\textbf{-bialg})^{\mathrm{op}} & \xrightarrow{(U_{B_\lambda})^{\mathrm{op}}} & (\Sigma\textbf{-alg})^{\mathrm{op}} & & \\ {\scriptstyle \overline{\Pi_2}}\uparrow & & \uparrow{\scriptstyle \Pi_2} & \nwarrow^{\Pi_1^{\mathrm{op}}} & \\ \kappa\textbf{-bialg} & \xrightarrow[U^{\widehat{\rho}_\kappa}]{} & \widehat{\zeta}\textbf{-coalg} & \cong & (\widehat{\zeta^\star}\textbf{-alg})^{\mathrm{op}} \end{array}$$

means that the top row of (20) commutes with (the opposites of) forgetful functors from λ-**bialg** and $\widehat{\zeta^\star}$-**alg** to Σ-**alg**. Thus we arrive at the conclusion that for any λ-bialgebra $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$, the interpretation of logic ρ on h is a carrier of a $\widehat{\zeta^\star}$-algebra and a Σ-algebra morphism from g. Note that neither ζ nor χ is mentioned in this conclusion, so the most useful way to state this is:

Theorem 4.6 *For any $S^{\mathrm{op}} \dashv T$, Σ, B, λ, L and ρ as above, if a Γ, ζ and $\kappa = \chi \circledast \lambda$ as above exist, then for any λ-bialgebra $\Sigma X \xrightarrow{g} X \xrightarrow{h} BX$, the interpretation of logic ρ on h is a Σ-algebra morphism from g.* □

When applied to initial λ-bialgebras, Theorem 2 of [15] is obtained.

5 Logical distributive laws over Set

Theorem 4.6 can be used to prove that a logical equivalence (defined by syntax L and semantics ρ) on a transition system (B-coalgebra) induced by a structural operational specification (defined by λ) is a congruence (with respect to syntax Σ). To use the theorem, one needs to find three additional ingredients: an endofunctor Γ and natural transformations ζ and χ such that (17) commutes. So far we have provided no intuitive meaning of these ingredients. This is the purpose of this section, where we restrict attention to the dual adjunction $\mathcal{C} = \mathcal{D} = \mathbf{Set}$, $S = T = 2^-$.

5.1 Distributive laws and predicate liftings

The search for Γ and ζ for a given Σ is entirely analogous to the search for modal logics L and ρ for a given B, as described in Section 3. One may therefore restrict attention to functors of the form

$$\Gamma = \coprod_{n\in\mathbb{N}} \Gamma_n \times (-)^n, \tag{21}$$

where $\Gamma_n \subseteq 2^{\Sigma 2^n}$. Recall that we may safely assume that L is of a similar form (7). This means that, once Γ with ζ were chosen, the last missing ingredient χ for Theorem 4.6 is a distributive law χ between polynomial functors. Such laws can be presented as systems of equations, as follows.

Suppose Γ and L are as in (21) and (7), presented by families of Σ- and B-modalities $(\Gamma_n)_{n\in\mathbb{N}}$ and $(L_n)_{n\in\mathbb{N}}$ respectively. Then a distributive law $\chi : L\Gamma \Longrightarrow \Gamma L$ is equivalent to a family of equations of the form:

$$\begin{gathered}\beta(\sigma_1(x_{11},\dots,x_{1m_1}),\dots,\sigma_n(x_{n1},\dots,x_{nm_n}))\\ =\\ \sigma(\beta_1(y_{11},\dots,y_{1l_1}),\dots,\beta_k(y_{k1},\dots,y_{kl_k})),\end{gathered} \tag{22}$$

where:

- $\beta \in L_n$, $\sigma_i \in \Gamma_{m_i}$, $\sigma \in \Gamma_k$ and $\beta_i \in L_{l_i}$,
- all variables x_{ij} are distinct,
- every variable y_{ij} occurs on the left side.

The latter two conditions determine a function $v : l \to m$, where $m = \sum_{i=1}^{n} m_i$ and $l = \sum_{i=1}^{k} l_i$ are arities of both sides of the equation.

To define a distributive law χ, the family must contain exactly one equation for each combination $\beta, \sigma_1, \dots, \sigma_n$ of a B-modality (of arity, say, n) and a

sequence of Σ-modalities (of length n).

We shall now formulate the condition (17), necessary for the application of Theorem 4.6, in terms of modalities and equations. To this end, first note that each $\beta, \sigma_1, \ldots, \sigma_n$ as on the left hand side of (22), defines a $B\Sigma$-modality of arity $m = \sum_{i=1}^{n} m_i$, (where $\sigma_i \in \Gamma_{m_i}$), which, following [27], will be denoted $\beta \circledcirc (\sigma_1, \ldots, \sigma_n) : B\Sigma(2^m) \to 2$. Moreover, the composite polynomial endofunctor $L\Gamma$ is represented by the collection of all such composite modalities, and the connection $\rho\Sigma^{\mathrm{op}} \circ L\zeta$ as in (17) (see also (10)) is obtained by copairing all the corresponding predicate liftings $(\beta \circledcirc (\sigma_1, \ldots, \sigma_n))^{\mathcal{Y}}$. Similarly one can define a lifting $\sigma \circledcirc (\beta_1, \ldots, \beta_k) : \Sigma B(2^l) \to 2$, from the right hand side of (22).

Since $L\Gamma$ is a polynomial functor, the condition (17) can be checked by cases, for each (left hand side of) equation (22). Each case amounts to checking that the following square of natural transformations commutes:

$$\begin{array}{ccc} (2^-)^m & \xRightarrow{(\beta \circledcirc (\sigma_1,\ldots,\sigma_n))^{\mathcal{Y}}} & 2^{B\Sigma -} \\ {\scriptstyle (2^-)^v}\Downarrow & & \Downarrow{\scriptstyle 2^{\lambda}} \\ (2^-)^l & \xRightarrow[(\sigma \circledcirc (\beta_1,\ldots,\beta_k))^{\mathcal{Y}}]{} & 2^{\Sigma B -}. \end{array}$$

By Yoneda lemma, this amounts to checking the equality of two ΣB-modalities of arity m:

$$(\beta \circledcirc (\sigma_1, \ldots, \sigma_n)) \circ \lambda_{2^m} = (\sigma \circledcirc (\beta_1, \ldots, \beta_k)) \circ \Sigma B(2^v) \tag{23}$$

This condition can be intuitively explained as follows. For a fixed set X, if predicates on X are substituted for variables $x_{i1}, \ldots, x_{im_i}$, then the expression $\sigma(x_{i1}, \ldots, x_{im_i})$ on the left hand side of (22) defines a predicate on ΣX; similarly, the entire left hand side defines a predicate on $B\Sigma X$ from a collection of m predicates on X. Further, the right hand side (together with the function $v : l \to m$ implicit in the equation) defines a predicate on ΣBX. Now the condition (23) means that the former predicate coincides with the latter when precomposed with λ.

5.2 A toy example

Consider Σ, B and λ as in (2) and (1) in Section 2. Consider also, as L and ρ, the trace equivalence defined in (6), represented by the collection of modalities given in (8). To apply Theorem 4.6 to infer the compositionality of trace equivalence for the language defined by (1), one needs to find a collection $(\Gamma_n)_{n \in \mathbb{N}}$ of Σ-modalities, and an collection of equations (22), such that the condition (23) holds for each equation.

As a first attempt, one might try the empty collection ($\Gamma_n = \emptyset$ for $n \in \mathbb{N}$), i.e., no Σ-modalities. There is only one left hand side of (22) to take care of:

the 0-ary B-modality $\top$. Unfortunately, however, there are no possible right hand sides of (22) at all, therefore no equation for $\top$ can be written.

To amend this, one can include an "always true" Σ-modality $\mathbb{T} : \Sigma 1 \to 2$ to Γ_0, formally defined by $\mathbb{T}(t) = \mathtt{tt}$ always. Then one can write an equation for $\top$:

$$\top = \mathbb{T} \tag{24}$$

and the condition (23) holds. Unfortunately now there are more left hand sides to take care of: no appropriate equation can be written for $\langle a \rangle(\mathbb{T})$.

The latter expression denotes a 0-ary $B\Sigma$-modality that, intuitively, checks whether some a-successor of a process exists. To express a corresponding (along λ) ΣB-modality, one may add, for each $a \in A$, a new unary Σ-modality $a \vee [\otimes] : \Sigma 2 \to 2$ to Γ_2, formally defined by: $a \vee [\otimes](t) = \mathtt{tt} \iff t \in \{a\ ,\ \mathtt{tt} \otimes \mathtt{tt}\}$, and write an equation:

$$\langle a \rangle \mathbb{T} = a \vee [\otimes](\langle a \rangle \top). \tag{25}$$

Intuitively, a process has an a-successor if and only if it is the process a or it is of the form $p \otimes q$ such that both p and q have a-successors. Formally, the condition (23) holds for this equation.

However, there is a slight problem here: formally, the right hand side of this equation is not of the form allowed in (22), as $\langle a \rangle \top$ is not a modality used in L. A principled solution to this problem would be to allow composite B-modalities on the right sides of equations; i.e., consider distributive laws $\chi : L\Gamma \Longrightarrow \Gamma L^*$, just as complex types of distributive laws are considered in the theory of SOS (see Section 2). Another solution is to simply add the missing (0-ary) B-modalities $\langle a \rangle \top$ to L_0 and proceed to find further equations. Changing a logic to prove its compositionality is an awkward step, but in this case it does not cause any serious harm, as the logical equivalence of the resulting logic is still trace equivalence. Formally, one then needs to provide suitable equations with 0-ary modalities $\langle a \rangle \top$ on the left hand side, but this is now straightforward:

$$\langle a \rangle \top = a \vee [\otimes](\langle a \rangle \top). \tag{26}$$

To complete the picture, one still needs to come up with equations for left hand sides such as $\langle a \rangle(b \vee [\otimes]x)$. This is solved by adding yet another, unary modality $[\otimes]$ to Γ_1, defined by $[\otimes](t) = \mathtt{tt} \iff t = \mathtt{tt} \otimes \mathtt{tt}$, with equations

$$\langle a \rangle(b \vee [\otimes]x)) = [\otimes]\langle a \rangle x, \qquad \langle a \rangle[\otimes]x = [\otimes]\langle a \rangle x.$$

These, together with (24–26), form a complete family of equations for our chosen Σ-modalities:

$$\Gamma_0 = \{\mathbb{T}\} \qquad \Gamma_1 = \{[\otimes]\} \cup \{a \vee [\otimes] \mid a \in A\} \qquad \Gamma_n = \emptyset \text{ for } n > 1$$

and the condition (23) holds for each equation, hence we can use Theorem 4.6 to conclude that trace equivalence is compositional for (1). The same result was used as an example in [15,17]; however, our crude understanding of Γ and χ there resulted in unnecessarily rich logical behaviours and complicated distributive laws.

5.3 Compositionality for expressive logics

An important question about the robustness of our approach to compositionality is whether Theorem 4.6 is a generalization of Proposition 2.1, i.e., whether it covers observational equivalence without any loss of generality. Under mild conditions (such as finitarity of B) studied in [16], observational equivalence on B-coalgebras is a logical equivalence for some logic (L, ρ) (such logic is called expressive). If this is the case, then the conclusion of Proposition 2.1 is a special case of the conclusion of Theorem 4.6. However, is there an expressive logic that satisfies the assumptions of Theorem 4.6?

We shall now give a partial positive answer to this question: we restrict attention to $\mathcal{C} = \mathcal{D} = \mathbf{Set}$ and $S = T = 2^-$, polynomial process syntax functors Σ, and finitary B that preserve finite sets. In the general case the question is left open.

For our special case, observational equivalence for B-coalgebras is defined by the expressive logic (L, ρ) presented by the collection of *all* finitary B-modalities, i.e., by $L_n = 2^{B2^n}$ for $n \in \mathbb{N}$. We shall now show that this logic satisfies the assumptions of Theorem 4.6 when one takes Γ and ζ presented by all Σ-modalities, i.e., $\Gamma_n = 2^{\Sigma 2^n}$ for $n \in \mathbb{N}$.

To this end we need, for every B-modality $\beta \in L_n$ and Σ-modalities $\sigma_1, \ldots, \sigma_n$, to present the ΣB-modality $(\beta \circledcirc (\sigma_1, \ldots, \sigma_n)) \circ \lambda_{2^m}$ (where m is the sum of arities of the σ_i) in the form

$$(\sigma \circledcirc (\beta_1, \ldots, \beta_k)) \circ \Sigma B 2^v \tag{27}$$

for some $\sigma \in \Gamma_k$ and $\beta_1, \ldots, \beta_k$ B-modalities with sum of arities l, and $v : l \to m$ a function between arities. Under our assumptions this can be done without any analysis of β, σ_i or λ, by the following result:

Proposition 5.1 *If Σ is polynomial and B preserves finite sets then every ΣB-modality of arity m can be decomposed as in (27).*

Proof. Let $\Sigma = \coprod_{i \in I} (-)^{n_i}$. Consider any $\gamma : \Sigma B(2^m) \to 2$. Put $k = |B(2^m)|$ (note that $B(2^m)$ is finite). For any $b \in B(2^m)$, define $\beta_b : B(2^m) \to 2$ by

$\beta_b(b') = \mathtt{tt} \iff b = b'$. Then define $\sigma : \Sigma(2^k) \to 2$ by:

$$\sigma(\iota_i(p_1, \ldots, p_{n_i})) = \mathtt{tt}$$
$$\iff$$
$$\forall b_1, \ldots, b_{n_i} \in B2^m.\ \big((\forall j = 1..n_i.\ p_j(b_j) = \mathtt{tt}) \Longrightarrow \gamma(\iota_i(b_1, \ldots, b_{n_i})) = \mathtt{tt}\big),$$

where $\iota_i : (-)^{n_i} \Longrightarrow \Sigma$ ranges over the coproduct injections into Σ. This gives a composite modality $\sigma \circledcirc (\beta_b)_{b \in B2^m}$ of arity $m \times k$, and it turns out that

$$\gamma = (\sigma \circledcirc (\beta_b)_{b \in B2^m}) \circ \Sigma B2^{\pi}$$

where $\pi : m \times k \to m$ is the evident projection. □

6 Future work

Unfortunately, both deficiencies that this paper aims at removing, still persist to some extent in the present formulation. On the abstract level, there clearly is a 2-categorical treatment of coalgebraic modal logic waiting to be discovered and combined with the one developed in [24] for bialgebras. Connections ρ and ζ are simply morphisms of endofunctors, just as λ and χ are endomorphisms on them; also sliced distributive laws $\chi \circledast \lambda$ are distributive law morphisms in the sense of [24]. There is clearly more structure in the story than currently explained.

On the concrete level, some more specific guidelines for finding suitable collections of Σ-modalities are much needed. Last but not least, more examples of logical distributive laws, and their relation to other work on SOS compositionality such as [6], need to be shown.

References

[1] L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, 2002.

[2] F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats.* PhD dissertation, CWI, Amsterdam, 2004.

[3] B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995.

[4] M. Bonsangue and A. Kurz. Duality for logics of transition systems. In *Proc. FOSSACS'05*, volume 3441 of *LNCS*, pages 455–469, 2005.

[5] M. Bonsangue and A. Kurz. Presenting functors by operations and equations. In *Proc. FOSSACS'06*, volume 3921 of *LNCS*, pages 172–186, 2006.

[6] W. J. Fokkink, R. J. van Glabbeek, and P. de Wind. Compositionality of Hennessy-Milner logic through structural operational semantics. In *Proc. FCT'03*, volume 2751 of *LNCS*, pages 412–422. Springer, 2003.

[7] J. F. Groote, M. Mousavi, and M. A. Reniers. A hierarchy of SOS rule formats. In *Proc. SOS'05*, 2005, pages 3–25. Elsevier, 2005.

[8] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[9] C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Information and Computation*, 145(2):107–152, 1998.

[10] B. Jacobs. Towards a duality result in the modal logic for coalgebras. In *Proc. CMCS 2000*, volume 33 of *ENTCS*, pages 160–195. Elsevier, 2000.

[11] B. Jacobs. Trace semantics for coalgebras. In *Proc. CMCS 2004*, volume 106 of *ENTCS*. Elsevier, 2004.

[12] B. Jacobs and A. Sokolova. Exemplaric expressivity of modal logics. *Journal of Logic and Computation*, 2009.

[13] P. T. Johnstone. Adjoint lifting theorems for categories of algebras. *Bull. London Math. Soc.*, 7:294–297, 1975.

[14] G. M. Kelly and R. Stret. Review of the elements of 2-categories. *Lecture Notes in Mathematics*, 420:75–103, 1974.

[15] B. Klin. Bialgebraic semantics and modal logic. In *Proc. LiCS'07*, pages 336–345. IEEE Computer Society Press, 2007.

[16] B. Klin. Coalgebraic modal logic beyond sets. In *Proc. MFPS 2007*, volume 173 of *ENTCS*, pages 177–201, 2007.

[17] B. Klin. Bialgebraic methods and modal logic in structural operational semantics. *Information and Computation*, 207:237–257, 2009.

[18] A. Kurz. Coalgebras and their logics. *ACM SIGACT News*, 37, 2006.

[19] A. Kurz and J. Rosický. The Goldblatt-Thomason theorem for coalgebras. In *Procs. CALCO 2007*, volume 4624 of *LNCS*, pages 342–355, 2007.

[20] M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theoretical Computer Science*, 327(1-2):135–154, 2004.

[21] S. Mac Lane. *Categories for the Working Mathematician*. Springer, second edition, 1998.

[22] D. Pattinson. Semantical principles in the modal logic of coalgebras. In *Proc. STACS 2001*, volume 2010 of *LNCS*. Springer, 2001.

[23] D. Pavlovic, M. Mislove, and J. B. Worrell. Testing semantics: connecting processes and process logics. In *Proc. AMAST'05*, volume 4019 of *LNCS*, pages 308–322. Springer, 2005.

[24] J. Power and H. Watanabe. Combining a monad and a comonad. *Theor. Comput. Sci.*, 280:137–162, 2002.

[25] V. R. Pratt. Chu spaces from the representational viewpoint. *Ann. Pure Appl. Logic*, 96:319–333, 1999.

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

[27] L. Schröder. Expressivity of coalgebraic modal logic: the limits and beyond. In *Proc. FOSSACS'05*, volume 3441 of *LNCS*, pages 470–484, 2005.

[28] D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, pages 280–291. IEEE Computer Society Press, 1997.

A Adjoint lifting

The following theorem is standard; a proof of it (more precisely, its dual) can be found in [9], see also [13].

Consider endofunctors B on $\mathcal{C}$ and B' on $\mathcal{C}'$, together with a functor $R : \mathcal{C} \to \mathcal{C}'$. A natural transformation $\alpha : RB \Longrightarrow B'R$ induces a functor $\overline{R} : B\textbf{-coalg} \to B'\textbf{-coalg}$ defined by:

$$\overline{R}(\, X \xrightarrow{h} BX \,) = \; RX \xrightarrow{Rh} RBX \xrightarrow{\alpha_X} B'RX \,. \tag{A.1}$$

Then, for the commuting diagram:

$$\begin{array}{ccc} B'\textbf{-coalg} & \xleftarrow{\overline{R}} & B\textbf{-coalg} \\ {\scriptstyle U_{B'}}\downarrow & & \downarrow{\scriptstyle U_B} \\ \mathcal{C}' & \xleftarrow[R]{} & \mathcal{C}, \end{array}$$

the following holds:

Proposition A.1 *If α is a natural isomorphism than a left adjoint $L \dashv R$ induces a left adjoint $\overline{L} \dashv \overline{R}$.*

Moreover, the adjunction $\overline{L} \dashv \overline{R}$ lifts $L \dashv R$ along the respective functors. In particular, since $U_{B'}$ reflects isomorphisms, this implies that if the unit of $L \dashv R$ is a natural isomorphism then so is the unit of $\overline{L} \dashv \overline{R}$.

B Proofs

B.1 Section 4.1: Not every endofunctor on $(\mathcal{D}{\downarrow}T)$ is sliced.

One important counterexample is the biextensional collapse construction on Chu spaces [25], seen as an endofunctor on $\mathbf{Chu}(\mathbf{Set}, 2) = (\mathbf{Set} \downarrow 2^-)$. For a simpler counterexample, consider $\mathcal{C} = \mathbf{Set}^{\mathrm{op}}$, $\mathcal{D} = \mathbf{Set}$ and $T = \mathrm{Id}$. Then $(\mathcal{D}{\downarrow}T) = \mathbf{Ar}(\mathbf{Set})$, the arrow category of $\mathbf{Set}$. Now consider an endofunctor $Q : \mathbf{Ar}(\mathbf{Set}) \to \mathbf{Ar}(\mathbf{Set})$ defined by:

- on objects, $Q(s : X \to Y) = m : Z \to Y$, where $X \xrightarrow{e} Z \xrightarrowtail{m} Y$ is the epi-mono factorization of s,
- on arrows, a pair $\langle f : X \to X', g : Y \to Y' \rangle$ such that

$$\begin{array}{ccc} X & \xrightarrow{s} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[s']{} & Y' \end{array}$$

commutes, is mapped to $\langle z, g\rangle$ as in the diagram:

$$\begin{array}{ccccc} X & \xrightarrow{e} & Z & \xleftarrow{m} & Y \\ {\scriptstyle f}\downarrow & & {\scriptstyle z}\downarrow & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[e']{} & Z' & \xleftarrow{m'} & Y', \end{array}$$

where z exists uniquely by the epi-mono factorization system of **Set**.

Functoriality of Q is ensured by the factorization system as well.

However, Q does not lift any functor on **Set** along Π_2, since it might give different results for different functions (objects in $\mathbf{Ar}(\mathbf{Set})$) even if they have the same domain.

B.2 Section 4.1: Proof of Prop. 4.2.

For any object $X \in \mathcal{C}$, consider the object $\langle TX, X, \mathrm{id}_X\rangle \in (\mathcal{D}{\downarrow}T)$ and define

$$\rho_X = \pi_3(K\,\langle TX, X, \mathrm{id}_X\rangle).$$

Then $\rho : LT \Longrightarrow TB^{op}$ is natural. Indeed, take any $f : X \to Y$ in $\mathcal{C}$. The square

$$\begin{array}{ccc} TY & = & TY \\ {\scriptstyle Tf}\downarrow & & \downarrow{\scriptstyle Tf} \\ TX & = & TX \end{array}$$

trivially commutes, hence

$$\langle Tf, f\rangle : \langle TY, Y, \mathrm{id}_Y\rangle \to \langle TX, X, \mathrm{id}_X\rangle$$

is a valid morphism in $(\mathcal{D}{\downarrow}T)$. But then also

$$K\,\langle Tf, f\rangle : K\,\langle TY, Y, \mathrm{id}_Y\rangle \to K\,\langle TX, X, \mathrm{id}_X\rangle$$

must be a valid morphism. Since K lifts L and B^{op}, there is $K\,\langle Tf, f\rangle = \langle LTf, Bf\rangle$ and by (9) the naturality square

$$\begin{array}{ccc} LTY & \xrightarrow{\rho_Y} & TBY \\ {\scriptstyle LTf}\downarrow & & \downarrow{\scriptstyle TBf} \\ LTX & \xrightarrow[\rho_X]{} & TBX \end{array}$$

commutes.

Moreover, $K = \widehat{\rho}$. To see this, it is enough to show, for any object $\langle \Phi, X, s\rangle$, that

$$\pi_3 K\,\langle \Phi, X, s\rangle = \pi_3 K\,\langle TX, X, \mathrm{id}_X\rangle \circ Ls.$$

To this end, notice that the square

$$\begin{array}{ccc} \Phi & \xrightarrow{s} & TX \\ {\scriptstyle s}\downarrow & & \Vert{\scriptstyle T\mathrm{id}_X} \\ TX & \xrightarrow[\mathrm{id}_{TX}]{} & TX \end{array}$$

obviously commutes, hence

$$\langle s, \mathrm{id}_X\rangle : \langle \Phi, X, s\rangle \to \langle TX, X, \mathrm{id}_X\rangle$$

is a valid morphism in $(\mathcal{D}{\downarrow}T)$. But then also

$$K\,\langle s, \mathrm{id}_X\rangle : K\,\langle \Phi, X, s\rangle \to K\,\langle TX, X, \mathrm{id}_X\rangle$$

must be a valid morphism. Since K lifts L and B^{op}, there is $K\,\langle s, \mathrm{id}_X\rangle = \langle Ls, \mathrm{id}_{BX}\rangle$ and by definition of morphisms in $(\mathcal{D}{\downarrow}T)$, the square

$$\begin{array}{ccc} L\Phi & \xrightarrow{\pi_3 K\langle \Phi, X, s\rangle} & TBX \\ {\scriptstyle Ls}\downarrow & & \Vert \\ LTX & \xrightarrow[\pi_3 K\langle TX, X, \mathrm{id}_X\rangle]{} & TBX \end{array}$$

commutes; but this is exactly the required equation. □

B.3 Section 4.1: Not every natural transformation between sliced endofunctors is sliced.

For a counterexample, take $\mathcal{C} = \mathbf{1}$, $\mathcal{D} = \mathbf{Set}$ and $T = C_2$ (the constant functor at a two-element set). Then $(\mathcal{D}{\downarrow}T) = \mathbf{Set}/2$ is the category of sets over a two-element set 2. Now consider endofunctors $K = \mathrm{Id}$ (the identity functor) and $K' = C_{\mathrm{id}_2}$ (the constant functor at id_2) on $(\mathcal{D}{\downarrow}T)$. It is easy to see that both functors are sliced.

Define $\kappa : K \Longrightarrow K'$ by $\kappa_{s:X\to 2} = s$. It is easy to see that this is well-defined as a morphism in $(\mathcal{D}{\downarrow}T)$; to show naturality, assume any $s : X \to 2$ and $r : Y \to 2$ and some $f : X \to Y$ such that $r \circ f = s$. The naturality square of κ at f is:

$$\begin{array}{ccc} s & \xrightarrow{f} & r \\ {\scriptstyle \kappa_s = s}\downarrow & & \downarrow{\scriptstyle \kappa_r = r} \\ \mathrm{id}_2 & \xrightarrow[\mathrm{id}_2]{} & \mathrm{id}_2 \end{array}$$

and this commutes immediately by the assumption on f.

However, the above κ is not sliced. To see this, take any two distinct functions $s, r : X \to 2$ for some set X. Then obviously $\kappa_s \neq \kappa_r$, therefore a purported $\alpha : \mathrm{Id} \Longrightarrow 2$ in **Set** cannot be defined on X.

B.4 Section 4.2: Proof of Proposition 4.5.

(12) shows how to define κ from α and β. For the other direction, for any $\kappa : \widehat{\rho} \Longrightarrow \widehat{\rho'}$, define

$$\alpha_\Phi = \Pi_1 \kappa_{\langle \Phi, S\Phi, \eta_\Phi \rangle} \qquad \beta_X = \Pi_2 \kappa_{\langle TX, X, \mathrm{id}_{TX} \rangle}$$

for any $X \in \mathcal{C}$ and $\Phi \in \mathcal{D}$, where $\eta : \mathrm{Id} \to TS^{\mathrm{op}}$ is the unit of $S^{\mathrm{op}} \dashv T$.

To check the naturality of α, for any $f : \Phi \to \Psi$ in $\mathcal{D}$ consider the first component of the naturality square of κ at $\langle f, S^{\mathrm{op}} f \rangle : \langle \Phi, S\Phi, \eta_\Phi \rangle \to \langle \Psi, S\Psi, \eta_\Psi \rangle$, which is a well-defined morphism in $(\mathcal{D} \downarrow T)$ by naturality of η. For the naturality of β, for any $g : X \to Y$ in $\mathcal{C}$ consider the second component of the naturality square of κ at $\langle Tg, g \rangle : \langle TY, Y, \mathrm{id}_{TY} \rangle \to \langle TX, X, \mathrm{id}_{TX} \rangle$, which is trivially a well-defined morphism in $(\mathcal{D} \downarrow T)$. The equation $T\beta^{op} \circ \rho = \rho' \circ \alpha T$ follows, for any given $X \in \mathcal{C}$, from the fact that the component of κ at $\langle TX, X, \mathrm{id}_{TX} \rangle$ is a well-defined morphism. Finally, it is straightforward to check that the construction of α and β from κ is mutually inverse with (12).

Coinduction in concurrent timed systems

Jan Komenda[1,2]

Institute of Mathematics, Czech Academy of Sciences, Brno, Czech Republic

Abstract

An important class of timed transition systems can be modeled by deterministic weighted automata, which are essentially partial Mealy automata, and their extensions using synchronous compositions defined over extended alphabets. From a coalgebraic viewpoint, behaviours of deterministic partial Mealy automata are causal and length preserving partial functions between finite and infinite sequences of inputs and outputs, called stream functionals. After a study of fundamental properties of functional stream calculus an application to the definition by coinduction of the synchronous product of stream functionals is proposed.

Keywords: deterministic weighted automata, Mealy automata, final coalgebra, synchronous product, coinduction

1 Introduction

Universal coalgebra as a general theory of (dynamical) systems offers definitions and proofs by coinduction [16], which are complementary to classical approaches based on induction and turned out to be valuable in simplifying the definitions and proofs of many concepts and properties that are hard or even impossible to formulate within the algebraic framework. The techniques borrowed from coalgebra have proven their usefullness in many areas of theoretical computer science (e.g. functional and object orienting programming), but also in control theory, in particular of discrete state transition systems that are called in control community discrete-event systems. The reference model for discrete-event systems are partial automata, which are coalgebras of a functor on the category of sets. They have been studied in [15] as the

[1] Email: komenda@math.cas.cz

[2] This work was supported by the Academy of Sciences of the Czech Republic, Inst. Research Plan No. AV0Z10190503 and by EU.ICT project DISC, N.224498.

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

model for control of discrete-event (dynamical) systems (DES) together with the partial automaton of (partial) languages as the final coalgebra. Purely logical DES in the form of partial automata have also been studied using coalgebraic techniques in [10].

Deterministic weighted automata and more generally deterministic transducers are typical instances of state transition structures that can easily be recasted as coalgebras of set functors. Actually, general nondeterministic weighted automata may be viewed as coalgebras as well, but it is difficult to put in use the corresponding final coalgebra [1], because the corresponding set functor involves powerset (even though sometimes only finite powerset is considered).

Deterministic (sequential) K-weighted automata with input alphabet A and weights in a semiring K are essentially partial Mealy automata [7], where the output alphabet is just replaced by a semiring K of weigths. The initial and output functions of Weighted automata (WA) are neglected or viewed as functions having values $0, 1 \in K$ defining initial and final states.

Mealy automata have been studied from a coalgebraic perspective in [18]. It has been shown that Mealy automata have final coalgebras, namely causal functions between streams (infinite sequences) over the input alphabet and streams over the output alphabet. This can be extended to the case of partial Mealy automata, which are Mealy automata with transition function that is only partially defined. Such a case is motivated by applications in control theory, where state-transition functions are partial functions corresponding e.g. to automata models of manufacturing systems. These are often represented by (timed) Petri nets [19] and can be translated into deterministic (weighted) automata using reachability graphs of (labelled) Petri nets, which are naturally partial automata.

It will be shown that the final coagebra of partial Mealy automata is formed by causal partially defined and length preserving functions between finite or infinite sequences over input alphabet and finite or infinite sequences over output set (semiring K).

WA model state transition systems with a quantitative information encoded by output values of transitions that can be e.g. cost of a transition, a timing information (like duration of executing a transition) or probability of a transition between given states. The underlying semiring is then typically the $(\mathbb{R} \cup \{\infty\}, min, +)$, $(\mathbb{R} \cup \{-\infty\}, max, +)$ or the probability semiring $(< 0, 1 >, +, .)$, respectively.

As for timed transition systems, there are two basic ways of representing complex timed systems with concurrency (i.e. simultaneous occurrence of events) by WA: use of nondeterminism and synchronous product constructs. The first one relies on nondeterminism. Indeed, it is well known that unlike logical automata, nondeterministic WA have significantly higher expressive

power compared to deterministic ones. More specifically, it is known from [9] that nondeterministic (max,+)-automata have a strong expressive power in terms of timed Petri nets: every 1-safe timed Petri net can be represented by a special (max,+) automaton, called heap model. The advantage of nondeterministic WA is that these are typically much smaller than their deterministic counterparts with the same behavior (if these happen to exist at all as finite WA). However, this approach is not easy to apply, because of problems with determinization and decidability issues [14].

The second way of modeling complex timed transition systems is using explicit product constructs. Partial Mealy automata with a suitable (number or interval based) semiring as an output set naturally model a simple class of timed transition systems: deterministic one clock timed automata or equivalently timed state graphs (machines). Timed state graphs are timed Petri nets, where no synchronization is allowed: every transition has exactly one upstream and one downstream place. The corresponding Mealy automata are simply formed using reachability graphs, where duration of a transition in the timed Petri net is exactly the output value of the associated transition in the Mealy automaton. In fact, such a Mealy automaton may be viewed as a one clock timed automaton [2], where the single clock is implicite and is replaced by the corresponding (exact) duration or interval duration from the underlying semiring (typically the so called (max,+) semiring $\mathbb{R}_{max} = (\mathbb{R} \cup \{-\infty\}, max, +)$ or the associated interval semiring). Such systems are intrinsically sequential, because the duration of consecutive events are simply added (the classical addition is the multiplication of $(\mathbb{R} \cup \{-\infty\}, max, +)$ to compute the execution times of event sequences (words). Therefore, explicit synchronous product is needed to model more complex timed behaviors (corresponding to multi-clock timed automata). Another reason why synchronous product of deterministic weighted automata is needed comes from applications, e.g. in manufacturing systems, where the underlying timed Petri nets models are formed by elementary timed state graphs that are composed using shared synchronization transitions. The overall system can then be modeled by the synchronous product of elementary timed state graphs (components) as in [19].

We have proposed a (truly) synchronous composition of deterministic (max,+)-automata based on tensor linear algebra and extended (multi-event set) alphabet in [12]. It turned out that there is no algebraic formula in terms of local (algebraic) behaviors (formal power series), but only using linear (automata) representations. In this paper a coalgebraic definition is given using coinductive definitions on stream functionals.

The paper is organized as follows. In Section 2 Mealy automata as coalgebras are recalled and partial Mealy automata are proposed. Final coalgebras of partial Mealy automata are studied and two theorems of functional stream calculus are stated. Section 3 is an introduction to deterministic (max,+) au-

tomata and their algebraic and coalgebraic behaviors. The notion of a timed language is recalled and compared to formal power series and causal stream functions. In section 4 coinductive definition of synchronous product of causal stream functions is proposed. An example is presented that illustrates the coalgebraic approach to concurrent timed systems. Finally, section 5 proposes a discussion and hints for future investigations.

2 Partial Mealy automata and deterministic weighted automata

In this section we recall from [18] Mealy automata as coalgebras and extend them to the case of partially defined transition function. A fundamental theorem of functional stream calculus is proposed that is the counterpart of fundamental theorem of (ordinary) stream calculus presented in [17]. Other properties of stream functions called stream functionals by analogy with mathematical analysis are stated.

Let A, K be arbitrary sets (typically finite and referred to as the set of inputs or events). The empty string will be denoted by λ. Further notation is $1 = \{\emptyset\}$ to denote a special one element set that will encode partiality of the transition function (when no transition is defined).

Definition 2.1 A partial Mealy automaton with inputs in A and outputs in K is the structure (S, t), where S is the set of states and the transition function is $t : S \to (1 + (K \times S))^A$. This function maps any state $s \in S$ a function $t(s) : A \to (1 + (K \times S))$ that associates to any input event A either a pair $\langle k, s' \rangle$ consisting of the new state and the output $k \in K$ or the symbol $\emptyset \in 1$. The latter case means that there is no transition from s to s' labeled by a and this is donoted by $s \stackrel{a}{\not\to}$.

Thus, Mealy automata with partially defined transition functions are coalgebras on the category Set of the functor $F : Set \to Set$ given by $F(S) = (1 + (K \times S))^A$.

The following notation, borrowed from [18], will be used: $s \stackrel{a|k}{\to} s'$ iff $t(s)(a) = \langle k, s' \rangle$. The fact that there is no transition labeled by a from s to any state is denoted by $s \stackrel{a}{\not\to}$. Since we work with deterministic Mealy machines this notation is justified and is equivalent to the absence transition labeled by a from s to any $s' \in S$.

Remark 2.2 If K is a semiring, i.e. K is endowed with addition and multiplication satisfying the semiring axioms, one may view partial Mealy automata as deterministic weighted automata. Typically, weighted automata are nondeterministic and have also quantitative initial and final functions that may represent cost, probability or duration (time) and associate to any state the

initial or final value in the corresponding semiring (i.e. $(R \cup \{\infty\}, min, +)$-semiring, probability semiring or $(R \cup \{-\infty\}, max, +)$-semiring), respectively. In other cases only logical initial and final functions are considered that determine simply initial or final states. The initial state and final states play no role in this study. It is implicitly assumed that any state is final and that there is exactly one initial state. Note that unlike weighted automata, where the fact that there is no transition from s to s' labeled by a is expressed by the zero value of the corresponding output from the semiring K, in our case the absence of transitions is encoded using special symbol $\emptyset$. Only later we will get rid of this symbol and represent partiality of transition functions by (only) partially defined stream functionals.

The basic cornerstones of coalgebras of partial Mealy automata are stated: homomorphisms and bisimulation relations. A *homomorphism* between two partial Mealy automata $S = (S, t)$ and $S' = (S', t')$ is a function $f : S \to S'$ such that for all $s \in S$ and $a \in A$: if $s \xrightarrow{a|b} s'$ then $f(s) \xrightarrow{a|b} f(s')$, which can be captured by the equality $F(f) \circ t = t' \circ f$ corresponding to the commutative diagram below:

$$\begin{array}{ccc} (1 + (K \times S))^A & \xleftarrow{\ t\ } & S \\ \Big\downarrow{\scriptstyle F(f)} & & \Big\downarrow{\scriptstyle f} \\ (1 + (K \times S'))^A & \xleftarrow{\ t'\ } & S' \end{array}$$

Definition 2.3 A *bisimulation* between two partial Mealy automata $S = (S, t)$ and $S' = (S', t')$ is a relation $R \subseteq S \times S'$ such that for all $s \in S$ and $s' \in S'$: if $\langle s, s' \rangle \in R$ then

(i) $\forall a \in A : \ s \xrightarrow{a|b} q \ \Rightarrow \ s' \xrightarrow{a|b'} q'$ such that $\langle q, q' \rangle \in R$, and $b = b'$, and

(ii) $\forall a \in A : \ s' \xrightarrow{a|b'} q' \ \Rightarrow \ s \xrightarrow{a|b} q$ such that $\langle q, q' \rangle \in R$, and $b = b'$.

(iii) $\forall a \in A$: $s \overset{a}{\not\to}$ iff $s' \overset{a}{\not\to}$

As usual, we write $s \sim s'$ whenever there exists a bisimulation R with $\langle s, s' \rangle \in R$. This relation is the union of all bisimulations, i.e. the greatest bisimulation also called bisimilarity.

2.1 Final partial Mealy machine

First we consider causal functions between infinite sequences over A and infinite sequences over $1 + K$. Only later these will be formulated in a different way, where partiality of the transition function on stream functions will be expressed without using special symbol $\emptyset \in 1$. Partial functions between fi-

nite or infinite sequences over A and finite or infinite sequences over K will be considered instead.

The set of all infinite sequences (streams) over a set A is denoted by A^ω. Similarly, the set of finite or infinite sequences (streams) over a set A is denoted by A^∞, i.e. $A^\infty = A^\omega \cup A^+$, where A^+ stands for $A^* \setminus \{\lambda\}$. The empty string λ is excluded from A^∞, because Mealy automata have no output on empty input (unlike Moore automata).

The elements of stream calculus as coinductive study of infinite sequences over a semiring K are first recalled from [17]. It relies on the fact that streams from K^ω together with the initial value (the initial output from K, also called head of the stream) and the stream derivative (also known as tail of the stream) form the final coalgebra $(K^\omega, \langle head, tail\rangle)$ of the set functor $F(S) = K \times S$. Formally, for $s = (s(0), s(1), s(2), s(3), \ldots) \in K^\omega$: $head(s) = s(0)$ and $tail(s) = s' = (s(1), s(2), s(3), \ldots)$.

The notion of stream derivative applies to both infinite and finite sequences. For a sequence $s = (s(0), s(1), s(2), s(3), \ldots, s(k)) \in A^+$ its stream derivative, denoted $s' \in A^*$, is defined by $s' = (s(1), s(2), s(3), \ldots, s(k))$. Otherwise stated, for $k = 0, 1, 2, \ldots$ $s'(k) = s(k+1)$. Obviously, s' does not preserve the length of finite sequences. For $a \in A$ and $\sigma = (\sigma(0), \sigma(1), \ldots, \sigma(k), \ldots) \in A^\infty$ the following notation is adopted: $a : \sigma = (a, \sigma(0), \sigma(1), \ldots)$.

The notion of causality [18] applies to functions between both finite and infinite sequences. f is causal means that for any $\sigma \in A^\infty$ the n-th element of the stream $f(\sigma) \in K^\infty$ depends only on the first n elements of $\sigma \in A^\infty$. Formally, $f : A^\infty \to K^\infty$ is *causal* if $\forall n \in \mathbb{N}$, $\sigma, \tau \in A^\infty$: $\forall i : i \leq n$: $\sigma(i) = \tau(i)$ then $f(\sigma)(n) = f(\tau)(n)$.

Unlike [18], where Mealy machines with complete transition functions are studied another property (that we call consistency of f) is required. Finally, $f : A^\infty \to (1+K)^\infty$ is called *consistent* if for any stream $\sigma \in A^\omega$ the sequence $f(\sigma) \in (1+K)^\omega$ has the property that the symbols $\emptyset$ must only be placed on the rightmost part of the stream. Formally, f is consistent if $\sigma \in A^\omega$: $f(\sigma)(k) = \emptyset$ then $f(\sigma)(n) = \emptyset$ for any $n > k$. The concept of consistency is close to prefix closedness of languages.

The initial output and functional stream derivative of f are defined in the same way as in [18]. Hence, $f[a] = f(a : \sigma)(0)$, which is well defined (independent of σ) due to causality. The functional stream derivative $f_a : A^\infty \to (1+K)^\infty$ is defined by $f_a(\sigma) = f(a : \sigma)'$. There seems to be a problem with defining stream derivative of a sequence of length 1. Fortunately, we only need the stream derivatives of finite sequences in the context of functional stream derivatives of [18], i.e. $f(a : \sigma)'$. Since $\sigma \in A^\infty$ is of length at least one, $a : \sigma$ is of length at least two, hence $f(a : \sigma)' \in (1 + K)^\infty$ is either of length at least one or is undefined.

It has been shown in [18] that causal functions from streams over A to

streams over K, where functional stream derivative plays the role of the transition function form final coalgebra of Mealy machines with complete transition functions.

Now we are ready to propose the following (universal) partial Mealy automaton.

Definition 2.4 Let us define the partial Mealy automaton $\mathcal{F} = (\mathcal{F}, t_{\mathcal{F}})$ with the carrier set $\mathcal{F} = \{f : A^\omega \to (1+K)^\omega \mid f \text{ is causal and consistent}\}$.

The first output and functional stream derivative endow $\mathcal{F}$ with partial Mealy automaton structure $(\mathcal{F}, t_{\mathcal{F}})$, where $t_{\mathcal{F}} : \mathcal{F} \to (1+(K\times\mathcal{F}))^A$ is defined by

$$t_{\mathcal{F}}(f)(a) = \begin{cases} \langle f[a], f_a\rangle & \text{if } f[a] \neq \emptyset \in 1, \\ \emptyset & \text{otherwise,} \end{cases}$$

The definition of derivative can be extended to strings using the classical chain rule: for $w \in A^*$ and $a \in A$: $(f_w)_a = f_{wa}$.

Below we point out that to any state of any partial Mealy automaton we can associate its behavior from $\mathcal{F}$ such that the mapping $f : S \to \mathcal{F}$ be a homomorphism of partial Mealy automata.

Proposition 2.5 *The partial Mealy automaton $(\mathcal{F}, t_{\mathcal{F}})$ is a final partial Mealy automaton: for every partial Mealy automaton (S,t) (with inputs in A and outputs in K), there exists a unique homomorphism $l : (S,t) \to (\mathcal{F}, t_{\mathcal{F}})$.*

Proof. For any Mealy automaton (S,t) we define a function $l : S \to \mathcal{F}$. It associates to a $s_0 \in S$ the function $l(s_0) : A^\omega \to (1+K)^\omega$ in the following way: for $\sigma \in A^\omega$ and $n \in \{0,1,2,\ldots\}$, the sequence of transitions corresponding to σ is considered (if it exists), i.e. $s_0 \overset{\sigma(0)|k_0}{\to} s_1 \overset{\sigma(1)|k_1}{\to} s_2 \ldots \overset{\sigma(n)|k_n}{\to} s_{n+1}$. We define in this case $l(s_0)(\sigma)(n) = k_n$. If there is no such a transition along the path labeled $\sigma(0)\ldots\sigma(n)$, then we put $l(s_0)(\sigma)(n) = \emptyset$. Otherwise stated, l maps any input sequence $\sigma \in A^\omega$ to the stream $(k_0, k_1, k_2, \ldots) \in (1+K)^\omega$ of outputs observed along this input starting in s_0. Then no transition possible starting from σ_m is expressed by putting special symbols encoding "empty" obervation: $k_l = \emptyset$ for $l \geq m$.

It is immediately seen that $l(s_0)$ is consistent, because clearly $l(s_0)(\sigma)(n) = \emptyset$ for any $n > m$ whenever $l(s_0)(\sigma)(m) = \emptyset$. It is not difficult to verify that $l(s_0)$ is causal and that l is a homomorphism, which is moreover a unique one (up to isomorphism). □

The stream function $l(s_0)$ above is called the (input-output) behavior of s_0. Final coalgebra $\mathcal{F}$ has the property that bisimulation on $\mathcal{F}$ implies (and henceforth is equivalent to) equality. This opens the possibility of proving equality of two causal and consistent stream functions $f, g \in \mathcal{F}$ by coinduction,

which amounts to showing that $f \sim g$: there exists a bisimulation relation $R \subseteq \mathcal{F} \times \mathcal{F}$ such that $\langle f, g\rangle \in R$. Since the transition function of $\mathcal{F}$ is defined using the tuple of the first output and functional stream derivative, the proof of R being a bisimulation (used further) consists of two steps. Firstly, it is shown that first outputs for any input coincide on all related pairs of stream functions, and secondly it is to be shown that the functional stream derivatives with respect to all input events are again related by R.

Now the final coalgebra $\mathcal{F}$ is formulated in an equivalent way, where $\emptyset$ is not needed, but both infinite and finite sequences of inputs and outputs are considered and the function between them must preserve their length. We will consider causal partial (partially defined) functions from finite or infinite sequences over A to finite or infinite sequences over K that are length preserving, i.e. finite sequences over A are mapped to finite sequences over K of the same length and infinite sequences over A are mapped to infinite sequences over K. It is easily seen that $\mathcal{F}$ is isomorphic to the following structure:

$$\mathcal{F}_\infty = \{f : A^\infty \to K^\infty | f \text{ length preserving and causal with } dom(f) \text{ prefix-closed}\}.$$

Note that the output set $1 + K$ is replaced by K and f is a partial function between finite or infinite sequences over input and output set. The consistency of $f : A^\omega \to (1 + K)^\omega$ enables to recast f as an element of $\mathcal{F}_\infty$.

It follows from the construction below. First, it is shown how to obtain from $f \in \mathcal{F}$ an element of $\mathcal{F}_\infty$. Since f is consistent, there are two possibilities: either for $\sigma \in A^\omega$ and any $n \in \mathbb{N}$ we have $f(\sigma)(n) \in K$, i.e. $f(\sigma)(k) \neq \emptyset$, or there is a $k \in \mathbb{N}$ such that $f(\sigma)(k) = \emptyset$. In the former case f is automatically an element of $\mathcal{F}_\infty$. In the latter case the consistency of f means that $f(\sigma)(n) = \emptyset$ for $n > k$. Then it is useless to evaluate f in the whole infinite sequence σ. It is then sufficient to consider only finite words like $\sigma(0)\sigma(1)\ldots\sigma(k) \in A^*$ and the corresponding finite words in the outputs $f(\sigma)(0).f(\sigma)(1)\ldots f(\sigma)(k) \in K^*$. It is clear that such a mapping f is only partial (as we forget the value of f on suffixes of $\sigma(0)\sigma(1)\ldots\sigma(k)$), and that f is length preserving. Let us not here that intuitively, $\sigma(0)\sigma(1)\ldots\sigma(k) \in A^*$ is in the (prefix-closed) language of the underlying partial Boolean automaton that forgets the outputs, while $\sigma(0)\sigma(1)\ldots\sigma(k+1)$ is already out of this prefix-closed language. It is immediately seen that $dom(f)$ is prefix-closed.

Conversely, to any length preserving and causal $f : A^\infty \to K^\infty$ with prefix-closed domain we can construct a causal and consistent mapping $f : A^\omega \to (1 + K)^\omega$. Indeed, it is simply sufficient to extend the maximal (wrt prefix-order) finite words for which f is defined to infinite words and complete the image sequences by symbols $\emptyset$ that are placed on the rightmost part of the streams $f(\sigma)$. Naturally, f is again causal and consistent.

An important observation is that for $f \in \mathcal{F}_\infty$ we have in fact $f[a] = f(a)(0)$

whenever f is defined for $a \in A$. Otherwise, $f[a]$ is undefined and there is no a-transition in $\mathcal{F}_\infty$ from f.

For each length preserving and causal function $f \in \mathcal{F}$ we define the initial (first) value (of output) corresponding to $a \in A$ simply by $f[a] = f(a)(0)$. Also, the functional stream derivative of f (with respect to input a) is defined as the function $f_a : A^\infty \to (1+K)^\infty$ given by $f_a(s) = f(a:s)'$ if it is defined. Clearly, f_a is again causal and preserves the length of s, because prefixig a finite sequence by a increases the length by 1 and f keeps the length, but the derivatives reduces it back to the original length. The transition function $t_{\mathcal{F}} : \mathcal{F} \to (1 + (K \times \mathcal{F}))^A$ is defined using the first output function and the functional stream derivative, similarly $t_{\mathcal{F}_\infty} : \mathcal{F}_\infty \to (1+(K \times \mathcal{F}_\infty))^A$ is defined by

$$t_{\mathcal{F}_\infty}(f)(a) = \begin{cases} \langle f[a], f_a \rangle & \text{if } f[a] \text{ is defined} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

2.2 Fundamental properties of stream functions

It is natural to call functions between streams by stream functionals to stress the analogy with functional analysis. First we recall other elementary concepts from stream calculus for streams over a semiring $K = (K, \oplus, \otimes, 0, 1)$. The constant stream corresponding to $r \in K$ is given by $[r] = (r, 0, \ldots)$. The notation $X = (0, 1, 0, \ldots)$ is important to describe any stream using constant streams, addition and multiplication. The addition (sum) of streams is defined using addition of K: For $\sigma, \tau \in K^\omega$:

$$(\sigma \oplus \tau)(n) = \sigma(n) \oplus \tau(n)$$

and Cauchy (convolution) multiplication of streams given by

$$(\sigma \otimes \tau)(n) = \bigoplus_{k=0}^{n} \sigma(k) \otimes \tau(n-k).$$

The symbol $\otimes$ for multiplication of streams is often left out as in the classical calculus, cf. $X f_{\sigma(0)}(\sigma')(0)$ below meaning $X \otimes f_{\sigma(0)}(\sigma')(0)$ etc. The n-th Cauchy power of X, i.e. $X^n = (0, \ldots, 0, 1, 0, \ldots)$ with 1 being placed on the $n+1$-st place. Finally the notation $\sigma^{(k)}$ is used to denote the k-th stream derivative of σ.

The following theorem is provided that is the counterpart of fundamental theorem of stream calculus in functional stream calculus. Fundamental theorem of stream functionals is stated in Theorem 2.6. Let us note that similarly as in the stream calculus the sum $\oplus$ is formal, although it is well defined, because there is only one element per component in $\oplus$ below.

Theorem 2.6 *For any $f \in \mathcal{F}$ and $\sigma = (\sigma(0), \sigma(1), \ldots, \sigma(k), \ldots) \in A^\omega$ we have:*

$$f(\sigma) = f(\sigma)(0) \oplus X f_{\sigma(0)}(\sigma')(0) \oplus \ldots X^k f_{\sigma(0)\ldots,\sigma(k-1)}(\omega^{(k)})(0) \oplus \ldots$$

or equivalently,

$$f(\sigma) = f[\sigma(0)] \oplus X f_{\sigma(0)}[\sigma(1)] \oplus \ldots X^k f_{\sigma(0)\ldots,\sigma(k-1)}[\sigma(k)] \oplus \ldots$$

Proof. It follows from the fundamental theorem of stream calculus, which states that for any $\sigma \in A^\omega$ it holds that $\sigma = \sigma(0) \oplus X.\sigma'$, which can be extended to $\sigma = \sigma(0) \oplus X\sigma'(0) \oplus X^2\sigma'(0) \oplus \ldots$ Similarly, it sufficies to show that $f(\sigma) = f(\sigma)(0) \oplus X f_{\sigma(0)}(\sigma')$, which is a direct consequence of the fundamental identity on K^ω for $f(\sigma)$: $f(\sigma) = f(\sigma)(0) \oplus X f(\sigma)'$. Indeed, $\sigma = \sigma(0) : \sigma'$, hence by definition of functional stream derivative we get: $f_{\sigma(0)}(\sigma') = f(\sigma(0) : \sigma')' = f(\sigma)'$. Hence, $f(\sigma) = f(\sigma)(0) \oplus X f_{\sigma(0)}(\sigma')$.

□

In the proof of Theorem 2.6 we did not make use of coinduction, but implicitly : it relies on fundamental theorem of stream calculus, which can be proven by coinduction. Let us also mention that similar fundamental theorem holds for functionals from $\mathcal{F}_\infty$. Now the set $\mathcal{F}_\infty$ is considered. In the next results partiality of the functionals from $\mathcal{F}_\infty$ is important. Functionals from $\mathcal{F}_\infty$ have interesting properties and some of them are proven below by coinduction on stream functions. Some of them are listed below. The fact that we can evaluate these functionals on finite words that are prefixes of (potentially) infinite words (streams) has interesting consequences. For instance, the lemma below.

Lemma 2.7 *For any $f \in \mathcal{F}_\infty$, $\omega \in A^\infty$, and $a \in A$: $f(a) : f_a(\omega) = f(a\omega)$. More generally, for any $u \in A^+$ and $\omega \in A^\infty$: $f(u) : f_u(\omega) = f(u\omega)$.*

Proof. First we stress that the $f(a) : f_a(\omega)$ is an element of K^∞. Hence, the equality can be shown by coinduction on streams [17] extended to finite and infinite sequences, which is the final coalgebra of partial stream automata given by the Set functor $F : S \to 1 + (K \times S)$. Put

$$R = \{\langle f(a) : f_a(\omega), f(a\omega)\rangle \cup \langle \sigma, \sigma\rangle \mid f \in \mathcal{F}_\infty,\ \sigma \in K^\infty,\ \text{and } a \in A.\}$$

This amounts to show that the heads (initial values) are the same and that the stream derivatives are also related by R. Firstly, $[f(a) : f_a(\omega)](0) = f(a)$ and $f(a\omega)(0) = f(a)(0) = f(a)$, because f is causal. Secondly, $\{f(a) : f_a(\omega)\}' = f_a(\omega)$ and $f(a\omega)' = f_a(\omega)$ from the very definition of functional stream derivative. It is also easy to see that $f(a) : f_a(\omega)$ can not make further

transition iff $f(a\omega)$ can not (namely iff $f(a)$ is undefined), which shows the partial stream counterpart of (iii) of Definition 2.3. □

An interesting and useful observation is that the first output function at first input $f[a] = f(a)(0)$ can be seen as a simple stream functional. Indeed, the initial output function can be seen as a particular partial stream functional defined by

$$f^{\infty}[a](\sigma) = \begin{cases} f[a] & \text{if } \sigma = a, \\ \text{undefined} & \text{otherwise: } \sigma \neq a, \end{cases}$$

The following concatenation like multiplication (denoted by $\odot$) of a stream functional g with this special stream functional $f^{\infty}[a]$ on the left, helps formulating another fundamental identity of functional stream calculus.

Definition 2.8 For any $f, g \in \mathcal{F}_{\infty}$, $\sigma = (\sigma(0) : \sigma') \in A^{\infty}$, and $a \in A$ we define

$$(f^{\infty}[a] \odot g)(\sigma(0) : \sigma') = \begin{cases} f(\sigma(0)) : g(\sigma') & \text{if } a = \sigma(0) \in dom(f), \\ \text{undefined} & \text{otherwise,} \end{cases}$$

Note that the multiplication on the right is just stream concatenation on K^{∞} following the notation of [17]. We make the convention that $f(a) : g(\sigma') = f(a)$ if $g(\sigma')$ is undefined. Addition on streams from K^{∞} induces addition on $\mathcal{F}_{\infty}$. Simply, one defines for $f_i \in \mathcal{F}_{\infty}$, $i \in I$: $(\bigoplus_{i\in I} f_i)(\sigma) = \bigoplus_{i\in I}(f_i(\sigma))$. Then we can write:

Theorem 2.9 *For any $f \in \mathcal{F}_{\infty}$ we have: $f = \bigoplus_{a\in A} f^{\infty}[a] \odot f_a$.*

Proof. It can be shown by coinduction on $\mathcal{F}_{\infty}$. We put

$$R = \{\langle \bigoplus_{a\in A} f^{\infty}[a] \odot f_a, f\rangle \cup \langle f, f\rangle \mid f \in \mathcal{F}_{\infty} \text{ and } a \in A\}.$$

Then R is a bisimulation on $\mathcal{F}_{\infty}$. Indeed, for any $b \in A$ we get $(f^{\infty}[a] \odot f_a)[b] = (f^{\infty}[a] \odot f_a)(b)(0) = (f^{\infty}[a](b)$. Let us observe that according to definition of $f^{\infty}[a]$ we have either $f^{\infty}[a](b) = f[b]$ or it is undefined, depending on $b = a$ or not. Hence, $\bigoplus_{a\in A}(f^{\infty}[a] \odot f_a)[b] = f[b]$ and the first outputs are equal for $\bigoplus_{a\in A}(f^{\infty}[a] \odot f_a)$ and f.

Now, let $(\bigoplus_{a\in A} f^{\infty}[a] \odot f_a \xrightarrow{b|k} f'$. Then $f'(\sigma) = (\bigoplus_{a\in A} f^{\infty}[a] \odot f_a)_b(\sigma) = \bigoplus_{a\in A}\{(f^{\infty}[a] \odot f_a)(b\sigma)\}' = \{f[b] : f_b(\sigma)\}' = f_b(\sigma)$. Again, $(f^{\infty}[a](b) = f[b]$ in case $b = a$ has been used. Finally, it is obvious that $\bigoplus_{a\in A} f^{\infty}[a] \odot f_a \overset{g}{\not\to}$ iff $f \overset{g}{\not\to}$ (namely iff $f[a]$ is undefined), which shows (iii) of Definition 2.3.

Hence, $\langle(\bigoplus_{a\in A} f^{\infty}[a] : f_a)_b, f_b\rangle \in R$, which was to be shown. □

This equality may be viewed as a functional stream counterpart of the fundamental theorem of (multivariable) formal power series (behaviors of Moore automata): $s = s(\lambda) + \sum_{a\in A} a.s_a$ for $s : A^* \to K$. Therefore, multiplying a stream functional f by the elementary functional given by $f[a]$, i.e. $f^\infty[a]$, in the sense of definition 2.8 can be seen as a functional counterpart of formal power series integration, which is given by $a\sigma$, i.e. $X\sigma$ for monovariable streams over $X = (0, 1, 0, \ldots)$. This is expressed in the Proposition below.

Proposition 2.10 *For any $f \in \mathcal{F}_\infty$ and $a \in A$: $(f^\infty[a] \odot f)_a = f$*

Proof. The equality is shown by coinduction on $\mathcal{F}_\infty$. Let

$$R = \{\langle (f^\infty[a] \odot f)_a, f\rangle \cup \langle f, f\rangle \mid f \in \mathcal{F} \text{ and } a \in A\}.$$

Then R is a bisimulation on $\mathcal{F}_\infty$. Indeed, for any $b \in A$ we get $(f^\infty[a]\odot f)_a[b] = \{(f^\infty[a] \odot f)(ab)\}'(0) = (f^\infty[a] \odot f)(ab)(1)=\{f(a) : f(b)\}(1) = f(b)$.

Now, let $(f^\infty[a] \odot f)_a \xrightarrow{b|k} f'$. Then $f'(\sigma) = (f^\infty[a] \odot f)_{ab}(\sigma) = \{(f^\infty[a] \odot f)(ab\sigma)\}'' = \{f(a) : f(b\sigma)\}'' = f(b\sigma)' = f_b(\sigma)$. Hence, $(f^\infty[a] \odot f)_{ab} = f_b$ and therefore $\langle (f^\infty[a] \odot f)_{ab}, f_b\rangle \in R$. Also, it is easy to see that $(f^\infty[a] \odot f)_a \overset{g}{\not\to}$ iff $f \overset{g}{\not\to}$ (namely iff $f[a]$ is undefined), which shows (iii) of Definition 2.3.

□

Remark 2.11 In section 4 we need a semiring structure on K in order to introduce the synchronous product operation on causal and length preserving functions that are behaviours of deterministic time-weighted automata.

Finally, let us mention that the behavior of Mealy automata are typically described in the literature as formal power series $f : A^+ \to K$. Still we prefer to work with $\mathcal{F}$, because as is shown in the next section, in the case of deterministic time-weighted automata coalgebraic behaviors are similar to timed languages from timed automata theory.

3 Deterministic weighted automata and timed languages

In this section deterministic (max,+) and interval automata without initial and final delays are considered and three representations of their behaviors are discussed: formal power series, timed languages and functions between finite or infinite sequences of inputs and outputs.

Let us start with the definition of deterministic K-weighted automata. Let $K = (K, \oplus, \otimes)$ be a semiring.

Definition 3.1 A deterministic K-weighted automaton over the input alphabet A and with weights in K is the Mealy automaton (S, t), where S is the set of states and the transition function is $t : S \to (1 + (K \times S))^A$.

Two important cases of semiring K are considered in this paper. A deterministic *(max,+)-automaton* is a deterministic K-weighted automaton with $K = \mathbb{R}_{max} = (R \cup \{-\infty\}, max, +)$. The zero element, i.e. $-\infty$ of $\mathbb{R}_{max}$ is denoted by ε in accordance with idempotent semiring notation [8] and the unit element is denoted by $e = 0$. A deterministic *interval automaton* is a deterministic K-weighted automaton with $K = \mathcal{I}_{max}^{max} = (\mathbb{R}\times\mathbb{R} \cup(-\infty, -\infty), \oplus, \otimes)$, where $\oplus$ is the componentwise maximum and $\otimes$ is the componentwise (conventional) addition.

At the first sight our deterministic K-weighted automata might seem very different from (non)deterministic K-weighted automata in algebra, which are defined by $G = (Q, A, \alpha, \mu, \beta)$, where Q is a finite set of states, A is the set of events, and the linear triple consists of $\alpha : Q \to K$, $t : Q \times A \times Q \to K$, and $\beta : Q \to K$, called input, transition, and output delays, respectively. Let us recall that t can be algebraically viewed as a collection of matrices

$$\mu : A \to K^{Q\times Q},\ \mu(a)_{q\,q'} \triangleq t(q, a, q').$$

Since the definition μ can be extended from $a \in A$ to $w \in A^*$ using the morphism property, i.e.

$$\mu(a_1 \dots a_n) = \mu(a_1) \otimes \dots \otimes \mu(a_n),$$

μ is often called a morphism matrix. Such a triple (α, μ, β) is called a linear representation of G. In this algebraic representation there is no need of using special symbols to express partiality of the transition function, because $t(q, a, q') = 0 \in K$ means there is no transition from q to q' labeled by a. The transition function associates to a state $q \in Q$, a discrete input $a \in A$ and a new state $q' \in Q$, an output value $t(q, a, q') \in K$ corresponding to the $a-$transition from q to q'.

As our K-weighted automata (viewed as coalgebras) are assumed to be deterministic, there is exactly one initial state and t is deterministic, i.e. $t : Q \times A \to (1 + K \times Q)$. Note that in this deterministic transition function t the set $\emptyset \in 1$ is needed to encode the partiality of the transition function, unlike the nondeterministic transition function, where the zero element of K, i.e. $0 \in K$ encodes the fact that there is no transition between two given states with a given label. It is immediately seen that this deterministic transition function can be recasted in the coalgebraic form in terms of the set functor $F(S) = (1 + (K \times S))^A$, cf. Definition 3.1.

Essentially, our attention is restricted to transition function, while initial and final weights (called delays in timed systems) are discarded, or at least α and β take their values in the Boolean subsemiring of K: e.g. $\forall q \in A : \alpha(q) \in \{0, 1\}$ meaning $\alpha(q) = 1$ iff q is the initial state. Initial and final state play no role in our study, because from a coalgebraic perspective any state can

play a role of an initial state in the sense that the behavior homomorphism $l : S \to \mathcal{F}$ evaluated in $s \in S$ gives the behavior of S, where s is the initial state.

Below it is assumed that $K = \mathbb{R}_{max}$. From an algebraic viewpoint, behaviors of timed systems are timed languages or formal power series. Below the notion of a timed language is recalled from the theory of timed automata [2].

Definition 3.2 A timed word s_t is a (finite or infinite) sequence over the alphabet $A \times \mathbb{R}$, i.e. $s_t \in (A \times \mathbb{R})^\infty$, where $(\sigma_1, t_1) \dots (\sigma_n, \sigma_n) \dots$ means that the execution time of an event a_i is achieved at time t_i, $i = 1, 2, \dots$. A timed language is a subset of timed words, i.e. $L_t \subseteq (A \times \mathbb{R})^*$.

It is easy to see the relationship between elements of the final coalgebra $\mathcal{F}$ or its equivalent presentation $\mathcal{F}_\infty$ from the previous section and timed languages. In fact, timed languages give the cumulated execution time of a sequence, which is the sum of durations of individual events in the sequence. This subsumes that the multiplication of the underlying output alphabet, which is the semiring $\mathbb{R}_{max}$ is the conventional addition. In particular, it means that the sequence of execution times $t_1 \dots t_n \dots$ from Definition 3.2 is nondecreasing.

On the contrary, for any partial, length preserving, and causal function $f : \mathcal{F}_\infty$ and $\sigma \in A^\infty$ the value $f(\sigma)$ gives the sequence of duration times of individual events from σ, which is naturally not nondecreasing in general. For a given $f \in \mathcal{F}_\infty$ it is easy to obtain the corresponding timed language by simply making the sum of the duration of consecutive events from the initial event up to a given one. For instance, the function $f : A^+ \to \mathbb{R}^+_{max}$ that maps the sequence a, b, c, b to the sequence $1, 2, 4, 3$ and is only defined on nonempty prefixes of a, b, c, d (e.g. a, b is mapped to the time sequence $1, 2$) corresponds to the finite timed language given by a single timed word $(a, 1)(b, 3)(c, 7)(b, 10)$. The timed words and such simple stream functionals are easy to obtain from one another. However, timed languages (subsets of timed words) are strictly more expressive than our stream functionals. This is natural, because timed words can express concurrent timed behaviors of general timed automata, while stream functionals are tailored to sequential (single clock) timed systems. Still it will be shown in section 4 that there is a class of concurrent timed behaviors that can be expressed by stream functionals using synchronous product based on extended alphabets.

Let us note that timed words are even closer to formal power series than stream functionals, but in general formal power series may hide some timing information. In the example above, the corresponding formal power series is $1a \oplus 3ab \oplus 7abc \oplus 10abcb$. However, for another formal power series $1a \oplus 7abc \oplus 10abcb$ there is no information about the execution time of ab, but only the one about abc is specified and we only know that ab is executed in time

interval $(1, 7)$.

But timed languages have no such a nice and rich structure as partial, length preserving, and causal functions from $\mathcal{F}_\infty$. This feature is important for our main goal: coinductive definition of synchronous product of deterministic weighted automata. Therefore, we only work with behaviors from $\mathcal{F}_\infty$ in the rest of this paper.

4 Behaviors of concurrent deterministic (max,+) automata

In this section the main result of this paper is presented: coinductive definition of synchronous product of behaviors of deterministic time-weighted automata, i.e. functions from $\mathcal{F}_\infty$, is proposed and discussed in detail.

Let us first recall the automaton based definition of synchronous product proposed in [12]. It is assumed that a distributed timed system is given by two deterministic (max,+)-automata. Let $G_1 = (S_1, t_1)$ and $G_2 = (S_2, t_2)$ be two (max,+)-automata defined over local alphabets A_1 and A_2. Then associated natural projections are $P_1 : (A_1 \cup A_2)^* \to A_1^*$ et $P_2 : (A_1 \cup A_2)^* \to A_2^*$. We also need the Boolean matrices associated to morphism matrices :

$$[B\mu(a)]_{ij} = \begin{cases} e = 0, & \text{if } [\mu(a)]_{ij} \neq \varepsilon \\ \varepsilon = -\infty, & \text{else} \end{cases}$$

In order to avoid heavy notation, $B\mu(a)$ is in the sequel denoted by $B(a)$. This notation can be extended to a (Boolean) morphism on words from $B : A^* \to \{0, -\infty\}^{n\times n}$ using morphism property $B(a_1 \dots a_n) = B(a_1) \dots B(a_n)$.

The synchronous composition of two (max,+)- automata is based on an extended alphabet composed of two types of events. Firstly, it includes all shared events. Secondly, it includes pairs of local sequences in between the synchronization event. Formally, $\mathcal{A} = (A_1 \cap A_2) \cup (A_1 \setminus A_2)^* \times (A_2 \setminus A_1)^*$. The problem is that the alphabet actually depends on the particular distributed timed system under consideration in the sense that not all pairs of local sequences (which would make the extended alphabet infinite) need to be included in $\mathcal{A}$. It suffices to include those pairs from $(A_1 \setminus A_2)^* \times (A_2 \setminus A_1)^*$ that are actually executed by the automata. In fact, we need to include in $\mathcal{A}$ exactly the pairs of maximal local strings (maximality is with respect to prefix order) in between two consecutive synchronization events. Note that this means that $\mathcal{A}$ can in general be infinite, but only in the case, where at least one local automaton G_1 or G_2 has no loops consisting of private events only $A_1 \setminus A_2$ or $A_2 \setminus A_1$, respectively. It is then natural to exclude this possibility by assuming that local (max,+)-automata G_1 and G_2 have no loops consisting of

private events. Otherwise stated, any loop in local subsystems must contain at least one shared (synchronization) event. In such a case the alphabet $\mathcal{A}$ for given distributed timed system is finite.

In order to define the synchronous product on behaviors from $\mathcal{F}_\infty$ it is necessary to use extended alphabet $\mathcal{A}$.

The Kronecker (tensor) product of matrices denoted by $\otimes^t$ is involved. If $A = (a_{ij})$ is a $m \times n$ matrix and B is a $p \times q$ matrix over K, then their *Kronecker (tensor) product* $A \otimes^t B$ is the $mp \times nq$ block matrix

$$(A \otimes^t B)_{ik,jl} = a_{ij} \otimes b_{kl}.$$

The parallel products of weighted automata à la A. Arnold [3] or P. Bucholtz [4] is not suitable for timed systems, because if these are applied to(max,+)-automata, their product as product of weighted automata remains a sequential model.

In the logical setting trace theory has been developed, where events that may occur simultaneously are related by independence relation. However, it is not clear how to extend the trace theory into the timed transition systems setting. Let us recall that classical composition of weighted automata [3] is simply given by tensor product of their linear representations. This corresponds to classical synchronous product of underlying Boolean automata, but the duration of a transition in the synchronous product is the product (i.e. conventional sum in $\mathbb{R}_{max}$)of the durations of participating transitions. This is not suitable for timed systems modeled by (max,+)-automata, unless there is no concurrency between events: e.g. the alphabets of subsystems are equal. Indeed, one would need to distinguish simultaneously executed events or strings and in the extreme case the duration of a global string is the sum of durations of corresponding (projected or local) strings. Therefore, definition below has been proposed, which corresponds to the intuition that in between two synchronizing transitions the local strings are executed in parallel, i.e. the duration of a string v of non shared events equals the maximum of durations of its projections $P_1(v)$ and $P_2(v)$ in local automata. A much simple coalgebraic counterpart of this definition will be given later in this section.

Definition 4.1 *Synchronous Composition* of (max,+)-automata $G_1 = (Q_1, A_1, \alpha_1, \mu_1, \beta_1)$ and $G_2 = (Q_2, A_2, \alpha_2, \mu_2, \beta_2)$, is the following interval automaton defined over the alphabet

$$\mathcal{A} = (A_1 \cap A_2) \cup [(A_1 \setminus A_2)^* \times (A_2 \setminus A_1))^*]$$

$$G_1 \| G_2 = \mathcal{G} = (Q_1 \times Q_2, \mathcal{A}, \alpha, \mu, \beta)$$

with $Q_1 \times Q_2$ set of states, $\mathcal{A}$ set of events, $\alpha = \alpha_1 \otimes^t \alpha_2$ the initial delay , $\mu : \mathcal{A}^* \to \mathbb{R}_{max}^{|Q|\times|Q|}$ the morphism matrix and $\beta = \beta_1 \otimes^t \beta_2$ final delay. The morphism matrix is defined by :

$$\mu(v) = \begin{cases} \mu_1(v) \otimes^t B_2(v) \oplus B_1(v) \otimes^t \mu_2(v), & \text{if } v = a \in A_1 \cap A_2 \\ \mu_1(P_1(v)) \otimes^t B_2(P_2(v)) \oplus B_1(P_1(v)) \otimes^t \mu_2(P_2(v)), & \text{if } v = (P_1(v), P_2(v)) \in \\ & (A_1 \setminus A_2)^* \times (A_2 \setminus A_1)^* \end{cases}$$

Let us now explain the intuition behind this seemingly complicated definition. The interval automata G_1 and G_2 are synchronized over the shared events set: $A_1 \cap A_2$, but in between two consecutive synchronizations the automata G_1 and G_2 are free to execute their respective private events belonging to $A \setminus (A_1 \cap A_2)$. These private events are represented by pairs of corresponding local strings $P_1(v) \in (A_1 \setminus A_2)^*$ and $P_2(v) \in (A_2 \setminus A_1)^*$. Not all events of $\mathcal{A}$ are needed in concrete synchronous products. In fact, only those pairs of local events are included that actually occur in local subsystems between two synchronizing events. The extended alphabet $\mathcal{A}$ is still potentially infinite, which is the main drawback of our approach. Fortunately, this may only happen if there are loops of private events in one of the subsystems and the subsystems are unable to synchronize. If this case is excluded a finite extended alphabet $\mathcal{A}$ can be found.

In [12] we could not find any algebraic definition compatible with this automata definition that would be based on formal power series $l_1 = l(G_1)$ and $l_1 = l(G_1)$, i.e. independent of the linear representation G_1 of l_1 and G_2 of l_2. A definition for behaviors is only possible if automata representations are fixed, but there is no algebraic definition independent of automata representation. Yet, the formula for behavior of the synchronous product from [12] is very complex and not practical to use, because for a given word over $A_1 \cup A_2$ it is the sum of a number of terms that is exponential in the number of synchronization events in this word. Still, even in the case of infinite alphabet $\mathcal{A}$, we have been able to compute the (algebraic) behavior of $G_1 \| G_2$ on finite words from $A = A_1 \cup A_2$. Indeed, any $w \in A^*$ can be decomposed as $w = v_0 a_1 v_1 \dots a_n v_n$, where $a_i \in A_1 \cap A_2,\ i = 1, \dots, n$ are shared (synchronization) events and $v_i \in (A \setminus (A_1 \cap A_2))^*,\ i = 0, \dots, n$ are sequences of private local events. For such a v_i the corresponding local strings in G_1 and G_2 are given by natural projections $P_1(v_i) \in A_1^*$ and $P_2(v_i) \in A_2^*$, respectively. This way, any word over distributed (global) event set A^* can be seen as the word $w = P_1(v_0) \times P_2(v_0) a_1 P_1(v_1) \times P_2(v_1) \dots a_n P_1(v_n 0) \times P_2(v_n)$ over the extended alphabet $\mathcal{A}$. The duration of private strings $v_i \in (A \setminus (A_1 \cap A_2))^*$ is simply given by the maximum of the durations of local strings $P_1(v)$ and $P_2(v)$.

In this paper such a definition for behaviors is given using finality of $\mathcal{F}_\infty$ formed by causal and length preserving functions between $\mathcal{A}^\infty$ and K^∞ with prefix closed domains. They are endowed by partial Mealy automaton struc-

ture as in section 2, where A is just replaced by the extended alphabet $\mathcal{A}$. The first output function of $(l_1 \| l_2)$ will be defined using first output functions of l_1 and l_2 extended to strings. The following concept is now needed. For $l_i \in \mathcal{F}_\infty$ over A_i and $v_i = a_1 \dots a_k \in A_i^+$ we define for $i = 1, 2$:

$$(l_i)[v_i] = (l_i)[a_1] \otimes (l_i)_{a_1}[a_2] \otimes \dots \otimes (l_i)_{a_1 \dots a_{k-1}}[a_k].$$

This is needed, because the whole local v_i string playing the role of $P_i(v)$ from definition below is executed in G_i in a sequential way: the duration of its execution is simply the usual sum, i.e. $\otimes$, of execution times of individual events $a_1, \dots a_k$ from v_i.

Definition 4.2 Define the following binary operation on $\mathcal{F}$ over $\mathcal{A}$: for $l_1, l_2 \in \mathcal{L}$ and $\forall v \in \mathcal{A}$:

$$\begin{aligned} (l_1 \| l_2)_v &= (l_1)_{P_1(v)} \| (l_2)_{P_2(v)} \text{ and} \\ (l_1 \| l_2)[v] &= l_1[P_1(v)] \otimes Bl_2[P_2(v)] \oplus Bl_1[P_1(v)] \otimes l_2[P_2(v)]. \end{aligned}$$

Note that equivalently one can write:

$$(1) \quad (l_1 \| l_2)[v] = \begin{cases} max(l_1[P_1(v)], l_2[P_2(v)]) & \text{if } l_i[P_i(v)] \neq \varepsilon \text{ for } i = 1, 2 \\ \varepsilon & \text{else, i.e. } \exists i = 1, 2 : l_i[P_i(v)] = \varepsilon \end{cases}$$

This definition is similar to the coinductive definition of synchronous product [15] of partial languages (behaviors of partial automata). Indeed, our functional input derivative for a shared event has the same form as the input derivative for languages. As for private events, the extended alphabet contains strings of these events (typically a finite number of them), but clearly our definition of functional input derivative is formally of the same form as the input derivative from the language definition [15] extended to strings. Namely, for partial languages $L_1 = (L_1^1, L_1^2)$, $L_2 = (L_2^1, L_2^2)$, and $w \in A^*$ we have in fact $(L_1 \| L_2)_w = (L_1)_{P_1(w)} \| (L_2)_{P_2(w)}$. Of course, the first output function is specific to the timed setting, but it is easy to understand from the equivalent form (1). This formal simplicity of Definition 4.2 is another advantage of the coalgebraic approach compared to the algebraic one, where it is only possible to give automata definitions. It seems that there is no algebraic formula in terms of local (algebraic) behaviors (formal power series), but only using linear (automata) representations. Coalgebraic framework makes it simpler due to the fact that final coalgebras are endowed with the same automaton structure as another automaton of a given functor. In fact, algebraic behaviors (formal power series) formal power series can also be endowed with a coalgebra structure [17], but only for the Moore automata functor. Maybe for this reason the algebraic definition of synchronous product is not so elegant and it seems to works on automata representations only: we could not find any definition for

formal power series.

Let us mention another two points. Firstly, in the definition of functional stream derivatives it is not necessary to distinguish the type of extended event. Secondly, the first output function can equivalently be written as follows, where we recall $\varepsilon = -\infty$ is the zero element of $\mathbb{R}_{max}$. The value ε in equation (1) can be viewed as undefined. In the special case with full synchronization, i.e. $A_1 = A_2$ there is no need for using extended alphabet, in fact in this case $\mathcal{A} = A_1 = A_2$. Note that for any $v = a \in \mathcal{A}$ we have in fact $P_1(v) = P_2(v) = a$.

Thus, we obtain for $l_1, l_2 \in \mathcal{L}$ and $\forall a \in A$:

$$(l_1 \| l_2)_a = (l_1)_a \| (l_2)_a$$

and $(l_1 \| l_2)[a] = l_1[a] \otimes Bl_2[a] \oplus Bl_1[a] \otimes l_2[a]$. Interestingly, synchronous product differs from sum or Hadamard product of two functionals (with obvious definitions) in the initial condition, which is different from both sum and the Hadamard product.

4.1 Example

In this section the coinductive definition of the last section is applied to a concrete example. We consider a simple distributed timed system consisting of two subsystems: (max,+)-automata G_1 and G_2 over the alphabets $A_1 = \{a, b, d\}$ and $A_2 = \{a, c\}$, respectively, drawn in figure 1. Their synchronous product is by Definition 4.1 the following (max,+)-automaton :

$$\mathbf{G_1} \| \mathbf{G_2} = \mathcal{G} = (\mathbf{Q_1} \times \mathbf{Q_2}, \mathcal{A}, \alpha, \mu, \beta),$$

where $Q_1 \times Q_2$ is the set of states,

$$\mathcal{A} = \{a, (be, c), (d, c)\} \subseteq (A_1 \cap A_2) \cup (A \setminus (A_1 \cap A_2))^*,$$

$\alpha = \alpha_1 \otimes^t \alpha_2$, $\beta = \beta_1 \otimes^t \beta_2$, and

$$\nu(v) = \begin{cases} \mu_1(a) \otimes^t B_2(a) \oplus B_1(a) \otimes^t \mu_2(a), & \text{if } v = a \in A_1 \cap A_2 \\ \mu_1(be) \otimes^t B_2(c) \oplus B_1(be) \otimes^t \mu_2(c), & \text{if } v = (be, c) \\ \mu_1(d) \otimes^t B_2(c) \oplus B_1(d) \otimes^t \mu_2(c), & \text{if } v = (d, c) \end{cases}$$

can easily be computed. The synchronous product $G_1 \| G_2$ is drawn in figure 1 on the right.

The behavior of the composed automaton is the synchronous product of behaviors of local components over $A_1 = \{a, b, d, e\}$ and $A_2 = \{a, c\}$, which are $l_1 : (A_1)^\infty \to (\mathbb{R}_{max})^\infty$ and $l_2 : (A_2)^\infty \to (\mathbb{R}_{max})^\infty$.

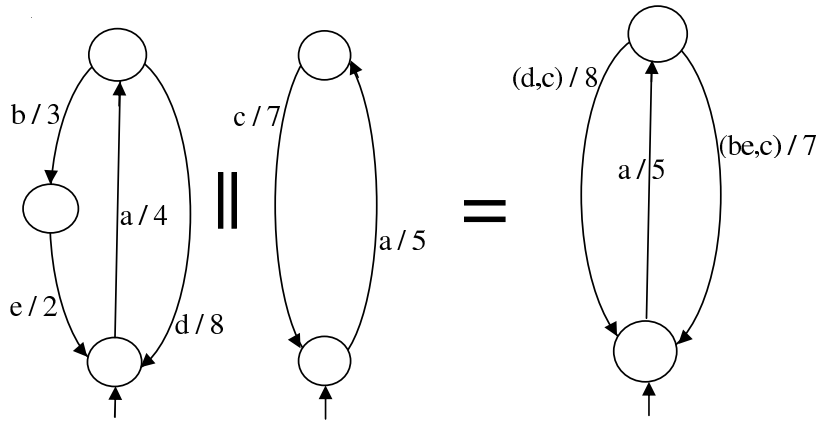


Fig. 1. G_1, G_2 and $G_1 \| G_2$

An important feature is that the extended alphabet is based on the underlying untimed partial automata and it includes only those sequences of private events that can occur in between two synchronization events (a). In our case these are sequences bec, bce, cbe (not to be distinguished one from another) represented by their projections to the local alphabets, i.e. (be, c). and sequences dc, cd represented by (d, c). Hence, the extended alphabet is $\mathcal{A} = \{a, (be, c), (d, c)\}$. Note that $P_1(be, c) = be \in A_1^*$, $P_2(be, c) = c \in A_2^*$, and similarly for (d, c). Hence, the differential equation for $(l_1 \| l_2)_v,\ v \in \mathcal{A}$ makes sense. The first output and derivative for $v = a$ are as follows:

$$(l_1 \| l_2)_a = (l_1)_a \| (l_2)_a \text{ and}$$
$$(l_1 \| l_2)[a] = l_1[a] \otimes Bl_2[a] \oplus Bl_1[a] \otimes l_2[a].$$

Let us mention that $(l_1 \| l_2)(a) = (l_1 \| l_2)(a)(0) = 5 = (l_1 \| l_2)[a]$, because $(l_1 \| l_2)$ is length preserving and $a \in A^\infty$ is of length 1. Now, according to the fundamental theorem of functional stream calculus we get

$$(l_1 \| l_2)(a(d, c)) = (l_1 \| l_2)(a(d, c))(0) \oplus (l_1 \| l_2)_a(d, c)(0).$$

Direct application of the formulas for derivative and first output function yields

$$(l_1 \| l_2)_a(dc) = ((l_1)_a \| (l_2)_a)(dc) = ((l_1)_a \| (l_2)_a)(dc)(0) = ((l_1)_a \| (l_2)_a)[dc]$$

$$= (l_1)_a[d] \otimes B(l_2)_a[c] \oplus B(l_1)_a[d] \otimes (l_2)_a[c] = (l_1)(ad)(1) \otimes B(l_2)(ac)(1) \oplus$$
$$B(l_1)(ad)(1) \otimes (l_2)(ac)(1) = 8 \otimes 0 \oplus 0 \times 7 = 8.$$

The second last equality follows from $f_a[d] = f_a(d)(0) = f(a : d)'(0) = f(ad)(1)$ for any $f \in \mathcal{F}_\infty$.

Similarly, we get $(l_1 \| l_2)(a(be, c)) = (l_1 \| l_2)(a(be, c))(0) \oplus (l_1 \| l_2)_a((be, c))(0)$, where $(l_1 \| l_2)_a(be, c) = \ldots = (l_1)(a(be))(1) \otimes B(l_2)(ac)(1) \oplus B(l_1)(a(be))(1) \otimes$

$(l_2)(ac)(1) = 5 \otimes 0 \oplus 0 \times 7 = 7$. These lines shows that the synchronous product is easy to compute and moreover if the composed automaton is drawn, its behavior is very intuitive and can directly be written down, cf. Figure 1.

The behavior functional $l_1 \| l_2 \in \mathcal{F}_\infty$ is only partially defined: essentially it is not defined for words that are outside the (prefix-closed) language of the underlying partial automaton and all infinite suffixes of such words. For instance, for $w = aa$ or $w = a^\omega$. On the other hand we have $(l_1 \| l_2)(a(d,c))^\omega = (5\ 8\ 5\ 8 \ldots) = (5\ 8)^\omega$.

The conclusion for this example is that both algebraic and coalgebraic framework can be used to compute the behaviors of synchronous product of (max,+)-automata. However, the coalgebraic approach does not use large matrices (but scalar behaviors), while the algebraic framework needs automata representations.

Remark 4.3 Two points are stressed. Firstly, our approach can be extended to more than two local components as described in [12], but it becomes quite complex. Secondly, synchronous product of interval automata, i.e. deterministic weighted automata with weights in $\mathcal{I}_{max}^{max}$, can be introduced in a similar way as the synchronous product of (max,+)-automata. Interval automata has been studied as B*ü*chi automata over interval based alphabets in [6] and their synchronous product are known as Product Interval Automata (PIA). The same construction based on extended event alphabet can be applied to interval automata and synchronous product of their behaviors can be defined by coinduction.

PIA correspond to an important class of timed automata, where the clocks are read (i.e. compared to constants in transition guards) and reset in a particular fashion: there are n clocks (one per component) and during a transition in a PIA only clocks that correspond to the components that are active in a transition are read and reset. This way the reading and reseting of clocks is compatible with the distributed event set structure. Thus, the usage of clocks can be completely avoided and PIA can be described by symbolic purely algebraic methods.

PIA are capable of modeling many interesting applications like asynchronous circuits. Hence, they represent a nice trade off between tractability (all fundamental problems are known to be decidable for this class of timed automata) and modeling power.

5 Concluding discussion

In this paper deterministic (sequential) weighted automata has been studied coalgebraically as partial Mealy automata. We have recasted their behaviors (causal and consistent stream functions) as partial, length preserving functions

(also called stream functionals) and extended basic results of stream calculus into functional stream calculus.

The main advantage of the coalgebraic approach is the possibility to use coinductive definitions and proofs that are known to be pertinent in many applications. Moreover, final coalgebra itself is endowed with the same structure as another coalgebra of a given functor. This helps defining operations on behaviors of state transition systems, e.g. streams, (partial) languages or (partial) stream functions, because these behaviors are seen as corresponding types of automata, e.g. stream automata, (partial) automata or Mealy automata. The corresponding definitions on automata are then simplified into coinductive definitions on behaviors.

In this work another application demonstrating power of coinductive definitions compared to definitions by induction is given: synchronous product of behaviors of deterministic weighted automata are defined by coinduction. The main advantage of our approach is that the composed automaton remains deterministic. On the other hand, the composed system has many states and decentralized approaches must be used in order to avoid the state explosion problem. Since our approach is compositional by construction, it is tailored to decentralized (component-wise) techniques.

Recently we have developed supervisory control theory for (max,+) automata that is applicable to nondeterministic (max,+) automata as well [11] (because determinism plays no essential role.) However, a major problem is that the resulting controller series computed within a behavioral (i.e. formal power series framework) need not be (max,+)-rational. Recall that a series is (max,+)- rational (respectively (min,+)- rational) if it is in the rational closure of series with finite supports, *i.e.* if it can be formed from polynomial series (*i.e.* those with finite support) by rational operation $\oplus$ (corresponding to max, respectively to min), $\otimes$, and the Kleene star. A notion close to that of a deterministic series is a *unambiguous series*, which is a series recognized by unambiguous automata, i.e. automata in which there is at most one successful path labeled by w for every word w. It is known from Lombardy and Sacharovitch [14] that the class of formal power series that are at the same time (max,+) and (min,+) rational coincides with unambiguous series. Moreover, for these families of series, the equality (and inequality) of series is proven to be decidable. Let us recall that sequentialization of weighted automata (i.e. their determinizing) and its decidability status is not known for formal power series over idempotent semirings (unlike the ring case, which is not so interesting for applications in distributed timed systems). The results of [14] show that essentially, beyond the class of deterministic series (i.e. those for which deterministic representations exist) and hence deterministic representations of the timed systems and their (control) specifications there is a little chance of obtaining a rational (i.e. finite state) controller automaton. Also, equality of

nondeterministic (max,+)-series is is known to be undecidable [13]. This is a major motivation for our study that consists in coding concurrent (non sequential) timed systems using synchronous product construct instead of using nondeterministic representations à la heap automata.

Among plans for further research, we plan to apply the coinductive definitions presented in this paper in the study of decentralized control (as in [10]) of distributed timed systems that are formed as synchronous compositions of sequential (i.e. one clock) systems. This would lead to an exponential saving of complexity of control synthesis compared to global control synthesis of distributed timed systems.

References

[1] J. Adámek, S. Milius and J. Velebil. *On coalgebra Based on Classes.* Theoretical Computer Science 316, pp.3-23, 2004.

[2] R. Alur and D. Dill. *The Theory of Timed Automata.* Theoretical Computer Science, 126:183-235, 1994.

[3] A. Arnold. *A. Arnold Finite Transition Systems. Semantics of Communicating Sytems*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[4] P. Buchholz, P. Kemper. *Weak Bisimulation for (max/+)-Automata and Related Models.* Journal of Automata, Languages and Combinatorics (2003) 8 (2), 187-218.

[5] S.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.

[6] D. D'Souza and P.S.Thiagarajan. *Product Interval Automata*, In Sadhana, Academy Proceedings in Engineering Sciences, Vol. 27, No. 2, Indian Academy of Sciences, pp. 181–208, 2002.

[7] S. Eilenberg. *Automata, Languages, and Machines*, Vol. A. Academic Press, New York, 1974.

[8] S. Gaubert. *Performance evaluation of (max,+) automata*, IEEE Trans. on Automatic Control, vol. 40(12), pp. 2014-2025, 1995.

[9] S. Gaubert and J. Mairesse. *Modeling and analysis of timed Petri nets using heaps of pieces.* IEEE Trans. on Automatic Control, vol. 44(4): 683-698, 1999.

[10] J. Komenda and J. H. van Schuppen: *Modular Control of Discrete-Event Systems with Coalgebra.* IEEE Transactions on Automatic Control, 53, N2 , pp. 447-460, 2008.

[11] J. Komenda, S. Lahaye, and J.-L. Boimond. *Supervisory Control of (max,+) automata: a behavioral approach.* Discrete Event Dynamic Systems, 19, N4 , pp. 525-549, Springer, 2009.

[12] J. Komenda, S. Lahaye, and J.-L. Boimond. *Le produit synchrone des automates (max,+).* Modélisation des Systèmes Réactifs (MSR09), Nantes, France, 2009. In JESA (Journal Européen des Systèmes Automatisés), vol. 43, pp.1033–1047, 2009.

[13] D. Krob. *The equality problem for rational series with multiplicities in the tropical semiring is undecidable.* Internat. J. Algebra Comput., 4, pp. 405 - 425, 1994.

[14] S. Lombardy and J. Mairesse. *Series which are both max-plus and min-plus rational are unambiguous*, RAIRO - Theoretical Informatics and Applications 40, pp. 1-14, 2006.

[15] J.J.M.M. Rutten. *Coalgebra, Concurrency, and Control. Research Report CWI*, SEN-R9921, Amsterdam, November 1999. Available also at `http://www.cwi.nl/~janr`.

[16] J.J.M.M. Rutten. *Universal Coalgebra: A Theory of Systems. Theoretical Computer Science* 249(1):3-80, 2000.

[17] J.J.M.M. Rutten. *Behavioural differential equations: a coinductive calculus of streams, automata, and power series.* Theoretical Computer Science Volume 308(1–3), pp. 1–53, 2003.

[18] J.J.M.M. Rutten. *Algebraic Specification and Coalgebraic Synthesis of Mealy Automata.* Proceedings FACS 2005, ENTCS Vol. 160, Elsevier, 2006, pp. 305-319.

[19] J. Sifakis and S. Yovine. *Compositional Specification of Timed Systems.* Invited paper in Proceedings of the 13th Annual Symp. on Theoretical Aspects of Computer Science, STACS'96, pp. 347-359, Springer LNCS 1046, February 1996.