Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 341 (2018) 261–276

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Iterated Covariant Powerset is not a Monad[1]

Bartek Klin[2]

Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
Warsaw, Poland

Julian Salamanca [3]

Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
Warsaw, Poland

Abstract

We prove that the double covariant powerset functor $\mathcal{PP}$ does not admit any monad structure. The same applies to the n-fold composition of $\mathcal{P}$ for any $n > 1$.

Keywords: monad, double covariant powerset, distributive law

1 Introduction

The categorical concept of a monad (see e.g. [19, Ch. VI]) has found multiple applications in mathematical foundations of programming science, and they have become an important design pattern in languages such as Haskell [12, 33] or Scala [27]. Depending on the context, monads can be viewed as abstract notions of computational effects [26, 34], or as collections to gather computed values [24], or as structures of values to be computed upon [3]. These perspectives are not mutually exclusive: for example, the (covariant) *powerset* monad $\mathcal{P}$ can be seen either as a very simple kind of unstructured collections, or as a carrier of nondeterminism as a computational effect.

An application where monads are viewed both as effects and as collections is the coalgebraic study of trace semantics of transition systems and automata (e.g. [6, 11,

[1] Research supported by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (ERC consolidator grant LIPA, agreement no. 683080).

[2] Email: `klin@mimuw.edu.pl`

[3] Email: `jsalamanca@mimuw.edu.pl`

https://doi.org/10.1016/j.entcs.2018.11.013
1571-0661/© 2018 The Author(s). Published by Elsevier B.V.

13,15,18,28,29]). There, a monad usually appears as a component of the type of behaviour of transition systems; more specifically, the component that represents the "type of branching" that systems exhibit, as opposed to the other component, the "type of transition", usually required to be simply an endofunctor. Typical monads used in this context are the powerset monad $\mathcal{P}$ or its finitary version the finite powerset monad $\mathcal{P}_{\mathrm{f}}$ for nondeterministic systems, and the probability distribution monad for probabilistic systems. One then resolves branching by interpreting systems in either the Kleisli or the Eilenberg-Moore category for the monad, either to collect behaviour traces of a system or to return a determinised system of a simpler type.

With this rough idea in mind, one naturally turns attention to alternating automata [5], which play a fundamental role in language theory and verification. To model them coalgebraically, it is natural to consider systems where the branching type is modeled by the *double* powerset functor. There are actually two functors that bear this name and act in the same way on objects: the double powerset $2^{2^{-}}$, which is the composition of the contravariant powerset with itself, and the double powerset $\mathcal{PP}$, which is the composition of the covariant powerset with itself. Coalgebras for the former functor (which, by the way, is a well-known monad), called neighbourhood frames [10], have a rather different behaviour from alternating automata. The use of double covariant powerset is more promising (see [14,15,17]), but before one applies, off-the-shelf style, the machinery of coalgebraic trace semantics to alternating automata, one needs to answer the question: is $\mathcal{PP}$ a monad?

This simple question does not seem to get a simple answer in the literature so far. Some authors avoid the question by looking for more laborious ways to deal with alternating behaviour (e.g. [15, Sec. 5.3] or [17, Ex. 4.5] or [14, Ex. 12] or [2]), which suggests that they do not expect an easy positive answer. Others give an explicit negative answer [31], but without any concrete evidence for it. On the other hand, Manes in [21, Ex. 2.12] proposed a specific monad structure on $\mathcal{PP}$, only to remark later [23] that the structure is flawed. (The same mistake was repeated in [22, Ex. 2.4.7] and independently by one of us in [16, Ex. 9], then noticed in [4, Sec. 7] and [17, Ex. 6.8].)

Our main contribution in this paper is a proof that the double covariant powerset $\mathcal{PP}$ cannot be endowed with *any* monad structure. The same applies to the double finite powerset $\mathcal{P}_{\mathrm{f}}\mathcal{P}_{\mathrm{f}}$. More generally, neither $\mathcal{P}^n$ nor $\mathcal{P}_{\mathrm{f}}^n$ (i.e. the n-fold composition of $\mathcal{P}$ or $\mathcal{P}_{\mathrm{f}}$) is a monad for any $n > 1$.

It is a standard result that a distributive law $\lambda : TS \Longrightarrow ST$ of a monad T over a monad S defines a monad structure on the composite functor ST. Our result therefore implies that there is no distributive law of the monad $\mathcal{P}$ over itself. Actually, we claim more: if we consider $\mathcal{P}$ merely as a pointed functor (i.e., as an endofunctor equipped with a unit natural transformation $\eta : \mathrm{Id} \Longrightarrow \mathcal{P}$, with no multiplication structure), there is still no distributive law of $\mathcal{P}$ over itself. Our proof of this resembles a previously known proof (credited to Plotkin in [32, Prop. 3.2]) that the probability distribution monad does not admit any distributive law over $\mathcal{P}$, and we formulate it in a way that generalises both cases: if a pointed functor T,

(1) preserves preimages and (2) admits what we call a *nontrivial idempotent term*, then there is no distributive law of T over $\mathcal{P}$ (both considered as pointed functors).

This contrasts with (but does not contradict) some known positive results about the existence of distributive laws. In [13, Sec. 4] it is proved that every functor that preserves weak pullbacks admits a distributive law over the monad $\mathcal{P}$. By [25, Thm. 2.9], every analytic functor has a distributive law over every commutative monad. ($\mathcal{P}$ is commutative, and although it is not analytic, it is easy to find an analytic functor that satisfies our conditions (1) and (2) above.) Finally, [15, Lem. 8] shows a very simple distributive law of any monad over the underlying functor of another monad, provided that the two monads are linked by a monad morphism. In all these works, distributive laws of the functor-over-monad or monad-over-functor shape are constructed, whereas we show the lack of distributive laws of pointed functors over pointed functors.

The structure of this paper is as follows. In Section 2 we prove a negative result about the existence of distributive laws of pointed functors. In Section 3, we prove that $\mathcal{PP}$ admits no monad structure, and in Section 4 we generalise that to $\mathcal{P}^n$ for any $n > 1$. In Section 5 we summarise some previous erroneous attempts to define a monad structure on $\mathcal{PP}$.

We are grateful to Mikołaj Bojańczyk for inspiring discussions and to Filippo Bonchi, Fabio Zanasi and Maaike Zwart for valuable comments. The first author thanks Joost Winter for pointing out the mistake in [16], and Jurriaan Rot for tracing it back to [22].

2 Distributive laws

Throughout this paper we work only in the category of sets and functions. A *monad* (T, η, μ) (see [19, Ch. VI] for more details) is an endofunctor T together with natural transformations $\eta : \mathrm{Id} \Longrightarrow T$ (the *unit*) and $\mu : TT \Longrightarrow T$ (the *multiplication*) such that the following diagrams commute:

$$\begin{tikzcd} T \arrow[r, Rightarrow, "\eta T"] \arrow[dr, equal] & TT \arrow[d, Rightarrow, "\mu"] \arrow[dl, phantom, "(\dagger)"] \arrow[dr, phantom, "(\ddagger)"] & T \arrow[l, Rightarrow, "T\eta"'] \arrow[dl, equal] \\ & T & \end{tikzcd} \qquad \begin{tikzcd} TTT \arrow[r, Rightarrow, "T\mu"] \arrow[d, Rightarrow, "\mu T"'] & TT \arrow[d, Rightarrow, "\mu"] \\ TT \arrow[r, Rightarrow, "\mu"'] & T. \end{tikzcd} \tag{1}$$

When no risk of confusion arises, we will denote such a monad simply by T.

The *(covariant) powerset* monad $\mathcal{P}$ is defined so that $\mathcal{P}X$ is the set of all subsets of X, and:

$$\eta_X(x) = \{x\}, \qquad \mu_X(\Phi) = \bigcup \Phi \qquad \text{for } x \in X, \Phi \subseteq \mathcal{P}X.$$

All results in this paper remain true when $\mathcal{P}$ is replaced by the *finite powerset monad* $\mathcal{P}_\mathrm{f}$, with the same proofs.

To simplify the presentation, we will only consider functors T that *preserve inclusions*, i.e., such that $X \subseteq Y$ implies $TX \subseteq TY$ and the inclusion function from

X to Y is mapped by T to the inclusion function from TX to TY. This assumption only matters in this section, and it could be dropped with little effort. The functor $\mathcal{P}$ obviously preserves inclusions.

A *pointed functor* (T, η) is an endofunctor T together with a unit natural transformation $\eta : \mathrm{Id} \Longrightarrow T$. Obviously every monad (and $\mathcal{P}$ in particular) is a pointed functor. A *distributive law* of a pointed functor (T, η^T) over a pointed functor (S, η^S) is a natural transformation

$$\lambda : TS \Longrightarrow ST$$

such that the following two *unit laws* hold:

$$\begin{tikzcd} T \arrow[r, Rightarrow, "T\eta^S"] \arrow[dr, Rightarrow, "\eta^S T"'] & TS \arrow[d, Rightarrow, "\lambda"'] & S \arrow[l, Rightarrow, "\eta^T S"'] \arrow[dl, Rightarrow, "S\eta^T"] \\ & ST & \end{tikzcd} . \tag{2}$$

If S and T are monads, λ becomes a distributive law of the monad T over the monad S if it satisfies two further axioms that involve the multiplication structures of S and T (see [1]).

The following definition is taken from [9] (where it is formulated more generally, without assumming that T preserves inclusions). Functors with the property below are also known as *taut* [20] or *semi-analytic* [30].

Definition 2.1 A functor T *preserves preimages* if for every function $f : X \to Y$, a subset $Z \subseteq Y$ and an element $t \in TX$,

$$\text{if} \quad Tf(t) \in TZ \quad \text{then} \quad t \in T(f^{-1}(Z)), \tag{3}$$

where $f^{-1}(Z) \subseteq X$ denotes the inverse image of Z along f.

If, additionally, $X \subseteq Y$ and $f : X \to Y$ is the inclusion function, the above property specialises to

$$TX \cap TZ \subseteq T(X \cap Z). \tag{4}$$

Most functors considered in coalgebra theory, including $\mathcal{P}$, preserve preimages (see [9] for a detailed study).

Definition 2.2 A *nontrivial idempotent term* for a pointed functor (T, η) is a natural transformation

$$\beta : \mathrm{Id} \times \mathrm{Id} \Longrightarrow T$$

such that:

- $\beta_X(x, x) = \eta_X(x)$ for each $x \in X$ (idempotence), and
- $\beta_{\{0,1\}}(0, 1) \notin T\{0\} \cup T\{1\}$ (non-triviality).

Example 2.3 The functor $\mathcal{P}$ admits a nontrivial idempotent term defined by:

$$\beta_X(x,y) = \{x,y\} \qquad \text{for } x,y \in X.$$

The probability distribution monad (called V in [32], but studied also in e.g. [8]) also admits a nontrivial idempotent term:

$$\beta_X(x,y) = \frac{1}{2}x + \frac{1}{2}y \qquad \text{for } x,y \in X.$$

Other examples of a nontrivial idempotent term include: the free (distributive) lattice monad ($\beta_X(x,y) = x \vee y$) and the free idempotent monoid monad ($\beta_X(x,y) = x \cdot y$).

Note that by the Yoneda Lemma, a natural transformation $\beta : \mathrm{Id} \times \mathrm{Id} \Longrightarrow T$ canonically corresponds to an element of $T2$. For $T = \mathcal{P}$, the β above corresponds to the element $2 \in T2$.

Theorem 2.4 *If a pointed functor (T,η) preserves preimages and admits a nontrivial idempotent term, then there is no distributive law of (T,η) over the pointed functor $\mathcal{P}$.*

Proof. Assume, towards a contradiction, that there is such distributive law $\lambda : T\mathcal{P} \Longrightarrow \mathcal{P}T$. Consider sets

$$A = \{a,b,c,d\} \qquad \text{and} \qquad U = \{u,v\}$$

and three functions $f,g,h : A \to U$ defined by:

$$\begin{aligned} f(a) = f(b) = u, \qquad && g(a) = g(c) = u, \qquad && h(a) = h(d) = u, \\ f(c) = f(d) = v, \qquad && g(b) = g(d) = v, \qquad && h(b) = h(c) = v. \end{aligned} \tag{5}$$

Consider the element

$$t = \beta_{\mathcal{P}A}(\{a,b\},\{c,d\}) \in T\mathcal{P}A$$

and analyse how the three naturality squares for f, g and h act on t:

$$\begin{tikzcd} T\mathcal{P}A \arrow[r, "T\mathcal{P}f"] \arrow[d, "\lambda_A"'] & T\mathcal{P}U \arrow[d, "\lambda_U"] \\ \mathcal{P}TA \arrow[r, "\mathcal{P}Tf"'] & \mathcal{P}TU \end{tikzcd} \qquad \begin{tikzcd} T\mathcal{P}A \arrow[r, "T\mathcal{P}g"] \arrow[d, "\lambda_A"'] & T\mathcal{P}U \arrow[d, "\lambda_U"] \\ \mathcal{P}TA \arrow[r, "\mathcal{P}Tg"'] & \mathcal{P}TU \end{tikzcd} \qquad \begin{tikzcd} T\mathcal{P}A \arrow[r, "T\mathcal{P}h"] \arrow[d, "\lambda_A"'] & T\mathcal{P}U \arrow[d, "\lambda_U"] \\ \mathcal{P}TA \arrow[r, "\mathcal{P}Th"'] & \mathcal{P}TU. \end{tikzcd}$$

Recall that $\mathcal{P}$ acts on functions by taking direct images, so in particular:

$$\begin{aligned} \mathcal{P}f\{a,b\} = \{u\} \qquad && \mathcal{P}g\{a,b\} = \mathcal{P}h\{a,b\} = \{u,v\}, \\ \mathcal{P}f\{c,d\} = \{v\} \qquad && \mathcal{P}g\{c,d\} = \mathcal{P}h\{c,d\} = \{u,v\}. \end{aligned}$$

By naturality and idempotence of β we get:

$$T\mathcal{P}g(t) = T\mathcal{P}h(t) = \beta_{\mathcal{P}U}(\{u,v\},\{u,v\}) = \eta_{\mathcal{P}U}\{u,v\}$$

hence, by a unit law for λ in (2):

$$\lambda_U(T\mathcal{P}g(t)) = \lambda_U(T\mathcal{P}h(t)) = \{\eta_U(u), \eta_U(v)\}.$$

By naturality squares for g and h we obtain:

$$\mathcal{P}Tg(\lambda_A(t)) = \mathcal{P}Th(\lambda_A(t)) = \{\eta_U(u), \eta_U(v)\}$$

which implies that $\lambda_A(t)$ is nonempty and

$$Tg(s), Th(s) \in \{\eta_U(u), \eta_U(v)\} \qquad \text{for every } s \in \lambda_A(t). \tag{6}$$

Now, if for example $Tg(s) = \eta_U(u) \in T\{u\}$ then, by (3) for $Z = \{u\}$, we obtain $s \in T\{a, c\}$. Applying the same reasoning to four cases in (6) we obtain:

$$s \in (T\{a, c\} \cup T\{b, d\}) \cap (T\{a, d\} \cup T\{b, c\}) \qquad \text{for every } s \in \lambda_A(t).$$

Distributing intersections over unions and using the intersection preservation property (4), we get:

$$s \in T\{a\} \cup T\{b\} \cup T\{c\} \cup T\{d\} \qquad \text{for every } s \in \lambda_A(t). \tag{7}$$

Now let us come back to the function f. By naturality of β we get:

$$T\mathcal{P}f(t) = \beta_{\mathcal{P}U}(\{u\}, \{v\})$$

hence, by the naturality square for f and by a unit law for λ in (2):

$$\mathcal{P}Tf(\lambda_A(t)) = \lambda_U(T\mathcal{P}f(t)) = \{\beta_U(u, v)\}.$$

This means that

$$Tf(s) = \beta_U(u, v) \qquad \text{for every } s \in \lambda_A(t).$$

But this, together with (7), contradicts the assumption that β is nontrivial. Indeed, if for example $s \in T\{a\}$ then $Tf(s) \in T\{u\}$ so $Tf(s)$ cannot be $\beta_U(u, v)$. □

This is essentially the same proof as in [32, Prop. 3.2] for the probability distribution monad taken as T, in that the same sets A, U and functions f, g, h are used there. Here we distilled assumptions so that the proof covers also the case of $T = \mathcal{P}$.

3 $\mathcal{P}\mathcal{P}$ is not a monad

Theorem 2.4 implies that there is no distributive law of the monad $\mathcal{P}$ over itself. Although such a law would be a natural way to define a monad structure on $\mathcal{P}\mathcal{P}$, this does not prove yet that such a monad structure does not exist. As shown in [1], composite monads that arise from distributive laws are of a special form, and not every monad is of that form in general.

Example 3.1 For any monoid $(M, e, \cdot)$, the functor $TX = M \times X$ is a monad with

$$\eta_X(x) = (e, x), \qquad \mu_X(g, h, x) = (g \cdot h, x) \qquad \text{for } x \in X, g, h \in M. \tag{8}$$

Let T be defined from the monoid $(\mathbb{Z}_2, 0, +)$ in this way. Now, any monoid on the set $\{0, 1, 2, 3\} \cong 2 \times 2$ defines a monad structure on the functor TT. Pick the commutative monoid M where $e = 0$ and $\cdot$ is addition with 3 playing the role of infinity (i.e., $x \cdot 3 = 3$ for every x and $2 \cdot 2 = 3$). From [1] we know that for every monad on TT that arises from a distributive law from T over T, $\eta T : T \Longrightarrow TT$ is a monad morphism. Both our T and TT arise from monoids as in (8), and monad morphisms between such monads correspond to homomorphisms between the corresponding monoids. However, the only monoid homomorphism from $\mathbb{Z}_2$ to our M is trivial and therefore non-injective. This gives a contradiction, since by (†) in (1) the transformation ηT must be pointwise injective. As a result, our M cannot be derived from any distributive law of T over T.

This example shows that sometimes monads on composite functors do not arise from distributive laws between those functors, so Theorem 2.4 does not quite answer our main question yet. However:

Theorem 3.2 *There is no monad structure on $\mathcal{PP}$.*

Proof. We use the same situation with sets A, U and functions $f, g, h : A \to U$ as in (5) in the proof of Theorem 2.4, but we analyse it some more. We remark that this idea is not unexpected. Indeed, in the recent work [7] it is stated that the original proof from [32], regarding the lack of distributive law of $\mathcal{P}$ over the probability distribution monad $\mathcal{D}$, can be modified to show that the composition $\mathcal{PD}$ admits no monad structure.

Denote $T = \mathcal{PP}$. Assume, towards contradiction, that there exist natural transformations $\eta : \mathrm{Id} \Longrightarrow T$ and $\mu : TT \Longrightarrow T$ that make (T, η, μ) a monad. By the Yoneda Lemma, we have that:

$$\mathrm{Nat}(\mathrm{Id}, T) = \mathrm{Nat}(\mathbf{Set}(1, \_), T) \cong T1 = \mathcal{PP}1 = \big\{\varnothing, \{\varnothing\}, \{1\}, \{\varnothing, 1\}\big\}.$$

Therefore, there are only four possible choices for η, which are defined for every $x \in X$ as:

$$\eta^{\clubsuit}_X(x) = \varnothing, \quad \eta^{\diamondsuit}_X(x) = \{\varnothing\}, \quad \eta^{\heartsuit}_X(x) = \big\{\{x\}\big\} \text{ and } \eta^{\spadesuit}_X(x) = \big\{\varnothing, \{x\}\big\}.$$

Note that $\eta^{\clubsuit}$ and $\eta^{\diamondsuit}$ cannot be the unit of such a monad since, by (†) in (1), every component of ηT must be injective.

Consider now the case of $\eta^{\heartsuit}$. For $A = \{a, b, c, d\}$, consider the element $\mathcal{S} \in TTA = \mathcal{PPPP}A$ given by:

$$\mathcal{S} = \bigg\{\Big\{\big\{\{a\}, \{b\}\big\}, \big\{\{c\}, \{d\}\big\}\Big\}\bigg\}. \tag{9}$$

Notice that, for $U = \{u,v\}$:

$$\mu_U\left(\left\{\left\{\{\{u\},\{v\}\}\right\}\right\}\right) = \{\{u,v\}\} \quad \text{and} \quad \mu_U\left(\left\{\left\{\{\{u\},\{v\}\}\right\}\right\}\right) = \{\{u\},\{v\}\}, \tag{10}$$

which follow from (‡) in (1) acting on $\{\{u,v\}\} \in TU$ and from (†) in (1) acting on $\{\{u\},\{v\}\} \in TU$, respectively. Now, consider the function $f : A \to U$ as defined in (5). By naturality of μ we have:

$$(Tf \circ \mu_A)(\mathcal{S}) = (\mu_U \circ TTf)(\mathcal{S}) \overset{(10)}{=} \{\{u,v\}\}.$$

Therefore, since T acts on functions by taking direct images, we have that:

$$\varnothing \neq \mu_A(\mathcal{S}) \subseteq \{\{a,b,c,d\},\{a,b,c\},\{a,b,d\},\{a,c,d\},\{b,c,d\},\{a,c\},\{a,d\},\{b,c\},\{b,d\}\}. \tag{11}$$

Now, consider the function $g : A \to U$ as defined in (5). By naturality of μ we have:

$$(Tg \circ \mu_A)(\mathcal{S}) = (\mu_U \circ TTg)(\mathcal{S}) \overset{(10)}{=} \{\{u\},\{v\}\}.$$

Together with (11), this implies that:

$$\mu_A(\mathcal{S}) = \{\{a,c\},\{b,d\}\}. \tag{12}$$

With this established, consider the function $h : A \to U$ as defined in (5). By naturality of μ we have:

$$\{\{u,v\}\} \overset{(12)}{=} (Th \circ \mu_A)(\mathcal{S}) = (\mu_U \circ TTh)(\mathcal{S}) \overset{(10)}{=} \{\{u\},\{v\}\},$$

which is a contradiction. Therefore, $\eta^{\heartsuit}$ cannot be the unit of such a monad.

Finally, consider the case of $\eta^{\spadesuit}$. For $A = \{a,b,c,d\}$ as before, consider the element $\mathcal{S} \in TTA = \mathcal{PPPP}A$ given by:

$$\mathcal{S} = \left\{\varnothing, \left\{\{\varnothing,\{a\},\{b\}\},\{\varnothing,\{c\},\{d\}\}\right\}\right\}. \tag{13}$$

By analogy to (10), for $U = \{u,v\}$ we have that:

$$\mu_U\left(\left\{\varnothing,\left\{\{\varnothing,\{u\}\},\{\varnothing,\{v\}\}\right\}\right\}\right) = \{\varnothing,\{u,v\}\} \quad \text{and}$$
$$\mu_U\left(\left\{\varnothing,\left\{\{\varnothing,\{u\},\{v\}\}\right\}\right\}\right) = \{\varnothing,\{u\},\{v\}\} \tag{14}$$

which follow from (‡) in (1) acting on $\{\varnothing,\{u,v\}\} \in TU$ and from (†) in (1) acting on $\{\varnothing,\{u\},\{v\}\} \in TU$, respectively. Now, consider the function $f : A \to U$ as defined in (5). By naturality of μ we have:

$$(Tf \circ \mu_A)(\mathcal{S}) = (\mu_U \circ TTf)(\mathcal{S}) \overset{(14)}{=} \{\varnothing,\{u,v\}\}.$$

Therefore, we have that:

$$\varnothing \neq \mu_A(\mathcal{S}) \subseteq \big\{\varnothing, \{a,b,c,d\}, \{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}, \{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}\big\}. \tag{15}$$

Now, consider the function $g : A \to U$ as defined in (5). By naturality of μ we have:

$$(Tg \circ \mu_A)(\mathcal{S}) = (\mu_U \circ TTg)(\mathcal{S}) \overset{(14)}{=} \big\{\varnothing, \{u\}, \{v\}\big\}.$$

Together with (15), this implies that:

$$\mu_A(\mathcal{S}) = \big\{\varnothing, \{a,c\}, \{b,d\}\big\}. \tag{16}$$

With this established, consider the function $h : A \to U$ as defined in (5). By naturality of μ we have:

$$\big\{\varnothing, \{u,v\}\big\} \overset{(16)}{=} (Th \circ \mu_A)(\mathcal{S}) = (\mu_U \circ TTh)(\mathcal{S}) \overset{(14)}{=} \big\{\varnothing, \{u\}, \{v\}\big\},$$

which is a contradiction. Therefore, $\eta^\spadesuit$ cannot be the unit of such a monad. This finishes the proof that there is no monad structure on $T = \mathcal{P}\mathcal{P}$. □

4 $\mathcal{P}^n$ is not a monad for $n > 1$

Intuitively, the cases of $\eta^\heartsuit$ and $\eta^\spadesuit$ in the proof of Theorem 3.2 clearly follow a similar pattern. To generalise the theorem to $\mathcal{P}^n$ (i.e. the n-fold composition of $\mathcal{P}$) we need to understand that pattern better, since candidates for a monad unit for $\mathcal{P}^n$ are more numerous and complicated, making ad-hoc reasoning impossible.

Theorem 4.1 *There is no monad structure on $\mathcal{P}^n$, for any $n > 1$.*

Proof. Denote $T = \mathcal{P}^n$. The case $n = 2$ was dealt with in Theorem 3.2, so we now assume that $n \geq 3$. Assume, towards contradiction, that there exist natural transformations $\eta : \mathrm{Id} \Longrightarrow T$ and $\mu : TT \Longrightarrow T$ that make the diagrams in (1) commute.

We use the same situation with sets $A = \{a,b,c,d\}$, $U = \{u,v\}$ and functions $f, g, h : A \to U$ as in (5) in the proof of Theorem 2.4.

Denote $1 = \{\star\}$. For a set X and elements $x \neq y \in X$, define $\gamma_{x,y} : \mathcal{P}^2 1 \to \mathcal{P}^2 X$ by:

$$\gamma_{x,y}(\varnothing) = \varnothing \qquad \gamma_{x,y}\Big(\big\{\{\star\}\big\}\Big) = \big\{\{x\}, \{y\}\big\}$$
$$\gamma_{x,y}\big(\{\varnothing\}\big) = \{\varnothing\} \qquad \gamma_{x,y}\Big(\big\{\varnothing, \{\star\}\big\}\Big) = \big\{\varnothing, \{x\}, \{y\}\big\}.$$

Then define $B_{x,y} \in \mathcal{P}^n X$ by:

$$B_{x,y} = (\mathcal{P}^{n-2}\gamma_{x,y})(\eta_1(\star)).$$

We shall use three objects defined by this formula: $B_{a,b}, B_{c,d} \in \mathcal{P}^n A$ and $B_{u,v} \in \mathcal{P}^n U$.

Furthermore, define a function $\sigma : \mathcal{P}1 \to \mathcal{P}^{n+1}A$ by:

$$\sigma(\emptyset) = \emptyset \qquad\qquad \sigma\big(\{\star\}\big) = \{B_{a,b}, B_{c,d}\}$$

and let $\mathcal{S} \in \mathcal{P}^{2n}A$ be defined by:

$$\mathcal{S} = (\mathcal{P}^{n-1}\sigma)(\eta_1(\star)). \tag{17}$$

Note that this definition of $\mathcal{S}$ coincides with (9) and (13) for the two particular η's considered in the proof of Theorem 3.2.

We will strive for contradiction by looking at the naturality squares for μ on maps f, g and h from (5), acting on the element $\mathcal{S}$:

$$\begin{tikzcd}
\mathcal{P}^{2n}A \arrow[r, "\mathcal{P}^{2n}f"] \arrow[d, "\mu_A"'] & \mathcal{P}^{2n}U \arrow[d, "\mu_U"] \\
\mathcal{P}^{n}A \arrow[r, "\mathcal{P}^{n}f"'] & \mathcal{P}^{n}U
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{P}^{2n}A \arrow[r, "\mathcal{P}^{2n}g"] \arrow[d, "\mu_A"'] & \mathcal{P}^{2n}U \arrow[d, "\mu_U"] \\
\mathcal{P}^{n}A \arrow[r, "\mathcal{P}^{n}g"'] & \mathcal{P}^{n}U
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{P}^{2n}A \arrow[r, "\mathcal{P}^{2n}h"] \arrow[d, "\mu_A"'] & \mathcal{P}^{2n}U \arrow[d, "\mu_U"] \\
\mathcal{P}^{n}A \arrow[r, "\mathcal{P}^{n}h"'] & \mathcal{P}^{n}U.
\end{tikzcd}$$

First, since $f(a) = f(b) = u$, the composition

$$\mathcal{P}^2 f \circ \gamma_{a,b} : \mathcal{P}^2 1 \to \mathcal{P}^2 U$$

is the function $\mathcal{P}^2(\star \mapsto u)$. As a result we obtain

$$\begin{aligned}(\mathcal{P}^n f)B_{a,b} = (\mathcal{P}^n f)\big((\mathcal{P}^{n-2}\gamma_{a,b})(\eta_1(\star))\big) &= \big(\mathcal{P}^{n-2}(\mathcal{P}^2 f \circ \gamma_{a,b})\big)(\eta_1(\star)) \\ &= \big(\mathcal{P}^n(\star \mapsto u)\big)(\eta_1(\star)) = \eta_U(u).\end{aligned} \tag{18}$$

By the same reasoning we get

$$(\mathcal{P}^n f)B_{c,d} = \eta_U(v). \tag{19}$$

On the other hand, since $g(a) = u$ and $g(b) = v$, the composition

$$\mathcal{P}^2 g \circ \gamma_{a,b} : \mathcal{P}^2 1 \to \mathcal{P}^2 U$$

is simply the function $\gamma_{u,v}$, and the same applies to h instead of g and/or c, d instead of a, b. As a result we obtain

$$(\mathcal{P}^n g)B_{a,b} = (\mathcal{P}^n g)B_{c,d} = (\mathcal{P}^n h)B_{a,b} = (\mathcal{P}^n h)B_{c,d} = B_{u,v}. \tag{20}$$

Now come back to the function σ. Using (18) and (19), the composition:

$$\mathcal{P}^{n+1} f \circ \sigma : \mathcal{P}1 \to \mathcal{P}^{n+1}U$$

is mapping $\emptyset$ to $\emptyset$ and $\{\star\}$ to $\{\eta_U(u), \eta_U(v)\}$. Define a function $\theta : \mathcal{P}1 \to \mathcal{P}U$ by:

$$\theta(\emptyset) = \emptyset \qquad\qquad \theta\big(\{\star\}\big) = \{u, v\}.$$

We have just shown that

$$\mathcal{P}^{n+1} f \circ \sigma = \mathcal{P}\eta_U \circ \theta. \tag{21}$$

We can therefore derive:

$$\begin{aligned}(\mathcal{P}^{2n} f)(\mathcal{S}) &\overset{(17)}{=} (\mathcal{P}^{2n} f)\big((\mathcal{P}^{n-1}\sigma)(\eta_1(\star))\big) = \big(\mathcal{P}^{n-1}(\mathcal{P}^{n+1} f \circ \sigma)\big)(\eta_1(\star)) \\ &\overset{(21)}{=} (\mathcal{P}^{n}\eta_U)\big((\mathcal{P}^{n-1}\theta)(\eta_1(\star))\big).\end{aligned}$$

By the unit law (‡) in (1) this implies that

$$\mu_U\big((\mathcal{P}^{2n} f)(\mathcal{S})\big) = (\mathcal{P}^{n-1}\theta)(\eta_1(\star)),$$

which by naturality of μ means that

$$(\mathcal{P}^{n} f)\big(\mu_A(\mathcal{S})\big) = (\mathcal{P}^{n-1}\theta)(\eta_1(\star)). \tag{22}$$

Let us now turn attention to the function g. By (20), the composition

$$\mathcal{P}^{n+1} g \circ \sigma : \mathcal{P}1 \to \mathcal{P}^{n+1} U$$

is mapping $\varnothing$ to $\varnothing$ and $\{\star\}$ to $\{B_{u,v}\}$. This means that:

$$\begin{aligned}(\mathcal{P}^{2n} g)(\mathcal{S}) &\overset{(17)}{=} (\mathcal{P}^{2n} g)\big((\mathcal{P}^{n-1}\sigma)(\eta_1(\star))\big) = \big(\mathcal{P}^{n-1}(\mathcal{P}^{n+1} g \circ \sigma)\big)(\eta_1(\star)) \\ &= \big(\mathcal{P}^{n}(\star \mapsto B_{u,v})\big)(\eta_1(\star)) = \eta_{\mathcal{P}^n U}(B_{u,v}).\end{aligned}$$

The same applies to h instead of g. This, by the unit law (†) in (1) and by naturality of μ, implies that

$$(\mathcal{P}^{n} g)\big(\mu_A(\mathcal{S})\big) = (\mathcal{P}^{n} h)\big(\mu_A(\mathcal{S})\big) = B_{u,v} = (\mathcal{P}^{n-2}\gamma_{u,v})(\eta_1(\star)). \tag{23}$$

From now on we will work with (22) and (23). We will first bring the exponent n in these equations down to 3, which will be enough to obtain a contradiction.

To this end, consider the natural transformation

$$\nu : \mathcal{P}^{n-2} \Longrightarrow \mathcal{P}$$

which is the standard multiplication of the monad $\mathcal{P}$, iterated in the obvious sense (remember that we assume that $n \geq 3$; for $n = 3$ we take $\nu = id_{\mathcal{P}}$). Note that this has nothing to do with the purported multiplication μ of $\mathcal{P}^n$. By naturality of ν and by (22) post-composed side-wise with $\nu_{\mathcal{P}^2 U}$, and denoting $Q = \nu_{\mathcal{P}^2 A}\big(\mu_A(\mathcal{S})\big)$, we obtain:

$$\begin{aligned}(\mathcal{P}^{3} f)(Q) = \nu_{\mathcal{P}^2 U}\Big((\mathcal{P}^{n} f)\big(\mu_A(\mathcal{S})\big)\Big) &\overset{(22)}{=} \nu_{\mathcal{P}^2 U}\big((\mathcal{P}^{n-1}\theta)(\eta_1(\star))\big) \\ &= (\mathcal{P}^{2}\theta)\Big(\nu_{\mathcal{P}^2 A}\big(\eta_1(\star)\big)\Big).\end{aligned} \tag{24}$$

Similarly, postcomposing both sides of (23) with $\nu_{\mathcal{P}^2 U}$, we obtain:

$$(\mathcal{P}^3 g)(Q) = \nu_{\mathcal{P}^2 U}\Big((\mathcal{P}^n g)\big(\mu_A(\mathcal{S})\big)\Big) \overset{(23)}{=} \nu_{\mathcal{P}^2 U}\big((\mathcal{P}^{n-2}\gamma_{u,v})(\eta_1(\star))\big) = (\mathcal{P}\gamma_{u,v})\Big(\nu_{\mathcal{P}^2 A}\big(\eta_1(\star)\big)\Big) \qquad (25)$$

and the same for h in place of g.

Note that the function $\gamma_{u,v}$ takes on only four distinct values. Denoting

$$\Gamma = \Big\{\varnothing, \{\varnothing\}, \big\{\{u\},\{v\}\big\}, \big\{\varnothing, \{u\}, \{v\}\big\}\Big\} \subseteq \mathcal{P}^2 U,$$

we may write the type of this function as $\gamma_{u,v} : \mathcal{P}^2 1 \to \Gamma$. Similarly, the function θ takes on only two distinct values, so $\mathcal{P}\theta$ can be given the type $\mathcal{P}\theta : \mathcal{P}^2 1 \to \Theta$ for:

$$\Theta = \Big\{\varnothing, \{\varnothing\}, \big\{\{u,v\}\big\}, \big\{\varnothing, \{u,v\}\big\}\Big\} \subseteq \mathcal{P}^2 U.$$

From (24) and (25) we infer:

$$(\mathcal{P}^3 f)(Q) \in \mathcal{P}\Theta, \qquad (\mathcal{P}^3 g)(Q) \in \mathcal{P}\Gamma, \qquad (\mathcal{P}^3 h)(Q) \in \mathcal{P}\Gamma.$$

This means that for every $R \in Q$ we have

$$(\mathcal{P}^2 f)(R) \in \Theta, \qquad (\mathcal{P}^2 g)(R) \in \Gamma, \qquad (\mathcal{P}^2 h)(R) \in \Gamma.$$

The first two of these conditions taken together imply, by using a similar reasoning as in the proof of Theorem 3.2, that:

$$R \in \Big\{\varnothing, \{\varnothing\}, \big\{\{a,c\},\{b,d\}\big\}, \big\{\varnothing, \{a,c\}, \{b,d\}\big\}\Big\}.$$

Then the extra condition $(\mathcal{P}^2 h)(R) \in \Gamma$ implies that $R \in \big\{\varnothing, \{\varnothing\}\big\}$, so we infer

$$Q \subseteq \big\{\varnothing, \{\varnothing\}\big\} = \mathcal{P}^2\varnothing \qquad \text{hence} \qquad Q \in \mathcal{P}^3\varnothing.$$

Recall that Q denotes $\nu_{P^2 A}\big(\mu_A(\mathcal{S})\big)$. By the way the natural transformation ν is defined, we get

$$\mu_A(\mathcal{S}) \in \mathcal{P}^n\varnothing.$$

Using (22) we infer

$$(\mathcal{P}^{n-1}\theta)(\eta_1(\star)) \in \mathcal{P}^n\varnothing$$

hence, by definition of θ and by how $\mathcal{P}^{n-1}$ acts on functions,

$$\eta_1(\star) \in \mathcal{P}^n\varnothing.$$

By naturality of η we get that for every set X and element $x \in X$

$$\eta_X(x) \in \mathcal{P}^n\varnothing.$$

But the set $\mathcal{P}^n\varnothing$ is finite and it does not depend on X, therefore for every X such that $|X| > |\mathcal{P}^n\varnothing|$ the function η_X cannot be injective. We arrive at a contradiction, since by the unit law (†) in (1) the function η_{TX} must be injective for every X. □

5 Mistakes

We shall now summarise different sources in which it has been mistakenly concluded that $\mathcal{P}\mathcal{P}$ is a monad.

5.1 *Mistake by Klin and Rot, 2015 [16]*

In [16, Ex. 9], it is claimed that a monad-over-monad distributive law $\lambda : \mathcal{P}\mathcal{P} \Longrightarrow \mathcal{P}\mathcal{P}$ could be formally defined by:

$$\lambda_X(\mathcal{A}) = \{\mathcal{P}g(\mathcal{A}) \mid g : \mathcal{A} \to X \text{ s.t. } g(A) \in A \text{ for each } A \in \mathcal{A}\} \qquad \text{for } \mathcal{A} \subseteq \mathcal{P}X. \quad (26)$$

In words, given a family $\mathcal{A}$ of subsets of X, λ_X returns the family of subsets obtained by picking a single element from every set in $\mathcal{A}$ in every possible way.

The mistake in this is that λ is not a natural transformation. This was noticed in [4, Sec. 7]; a variant of the argument, pointed out to us by J. Winter, is the following. Consider

$$X = \{a,b,c\} \qquad Y = \{d,e\} \qquad f(a) = f(b) = d \qquad f(c) = e.$$

The naturality square for $f : X \to Y$ does not commute, as shown here:

$$\begin{tikzcd}
\{\{a,c\},\{b,c\}\} \arrow[rrr, mapsto] \arrow[ddd, mapsto] \arrow[dr, dotted, no head, "\in"] & & & \{\{d,e\}\} \arrow[dd, mapsto] \\
& \mathcal{P}^2X \arrow[r, "\mathcal{P}^2 f"] \arrow[d, "\lambda_X"'] & \mathcal{P}^2Y \arrow[d, "\lambda_Y"] \arrow[ur, dotted, no head, "\ni"] & \\
& \mathcal{P}^2X \arrow[r, "\mathcal{P}^2 f"] & \mathcal{P}^2Y \arrow[r, dotted, no head, "\ni"] \arrow[d, dotted, no head, "\in"] & \{\{d\},\{e\}\} \\
\{\{a,b\},\{a,c\},\{b,c\},\{c\}\} \arrow[rr, mapsto] \arrow[ur, dotted, no head, "\in"] & & \{\{d\},\{d,e\},\{e\}\}. \arrow[ur, dotted, no head, "\neq"] &
\end{tikzcd}$$

5.2 *Mistake by Manes and Mulry, 2003-07 [21, 22]*

In [22, Ex. 2.4.7], a "distributive law" of $\mathcal{P}$ over $\mathcal{P}$ is defined by:

$$\lambda_X(\mathcal{A}) = \left\{ \{a_A \mid A \in \mathcal{A}\} \;\middle|\; (a_A) \in \prod_{A \in \mathcal{A}} A \right\} \qquad \text{for } \mathcal{A} \subseteq \mathcal{P}X. \quad (27)$$

It is not difficult to see that this definition is equivalent to (26), and so it does not define a natural transformation. In [22] naturality of λ is actually inferred from the naturality of the unit and multiplication of a purported monad $\mathcal{P}\mathcal{P}$ claimed there.

The reader of [22] is referred to [21, pages 76–79] for a proof that $\mathcal{P}\mathcal{P}$ is a monad. There, the monad is defined in terms of a Kleisli triple $(\mathcal{P}\mathcal{P}, \eta, (-)^{\#})$ and the monad multiplication μ is derived from that in the usual way (see [21, Prop. 2.14]).

Given a function $f : X \to \mathcal{P}\mathcal{P}Y$, a function $f^{\#} : \mathcal{P}\mathcal{P}X \to \mathcal{P}\mathcal{P}Y$ is defined in [21] by:

$$f^{\#}(\mathcal{A}) = \left\{ \bigcup_{x\in A} B_x \;\middle|\; A \in \mathcal{A},\ \forall x \in A. B_x \in f(x) \right\}.$$

This is equivalent to saying that

$$B \in f^{\#}(\mathcal{A}) \iff \exists A \in \mathcal{A}. \exists (B_x \in f(x))_{x\in A}.\ B = \bigcup_{x\in A} B_x \tag{28}$$

for $\mathcal{A} \subseteq \mathcal{P}X$. Here, the second existential quantifier means that "there exists a family $(B_x)_{x\in A}$ such that every B_x belongs to $f(x)$".

Viewing this $(-)^{\#}$ as a Kleisli extension and the obvious $\eta_X(x) = \{\{x\}\}$ as the unit, the usual construction gives a "monad" structure on $\mathcal{P}\mathcal{P}$ as described in [22]. Since that structure is wrong, one expects problems with the Kleisli triple, and indeed the axiom:

$$(g^{\#} \circ f)^{\#} = g^{\#} \circ f^{\#} \qquad \text{for } f : X \to \mathcal{P}\mathcal{P}Y,\ g : Y \to \mathcal{P}\mathcal{P}Z \tag{29}$$

fails.

In [21], on pages 78–79, a proof of the axiom is attempted. The left-hand side is rewritten as:

$$C \in (g^{\#} \circ f)^{\#}(\mathcal{A}) \iff \exists A \in \mathcal{A}. \exists (B_x \in f(x))_{x\in A}. \exists (C_{x,y} \in g(y))_{x\in A, y\in B_x}. C = \bigcup_{x\in A} \bigcup_{y\in B_x} C_{x,y}$$

and this transformation is correct. The right-hand side is first rewritten as:

$$\begin{aligned} C \in (g^{\#}(f^{\#}(\mathcal{A})) &\iff \exists B \in f^{\#}(\mathcal{A}). \exists (C_y \in g(y))_{y\in B}.\ C = \bigcup_{y\in B} C_y \\ &\iff \exists A \in \mathcal{A}. \exists (B_x \in f(x))_{x\in A}. \exists (C_y \in g(y))_{x\in A, y\in \bigcup_{x\in A} B_x}. C = \bigcup_{y\in \bigcup_{x\in A} B_x} C_y \end{aligned}$$

and this is also correct. However, in the last equivalence on page 78, this is then equated to the left-hand side, and this is incorrect. Intuitively, looking at the third existential quantifiers on both sides above, the equality may not hold if the family $(B_x)_{x\in A}$ contains some overlapping sets.

Indeed, the axiom (29) fails for the following data:

$$X = \{1, 2\} \qquad Y = \{*\} \qquad Z = \{a, b\}$$
$$f(1) = f(2) = \{\{*\}\} \qquad g(*) = \{\{a\}, \{b\}\}.$$

To see this, calculate from (28)

$$B \in f^{\#}\big(\{\{1, 2\}\}\big) \iff \exists A = \{1, 2\}. \exists B_1, B_2 = \{*\}. B = B_1 \cup B_2$$

so $f^{\#}\Big(\big\{\{1,2\}\big\}\Big) = \big\{\{*\}\big\}$. Further, again from (28):

$$C \in g^{\#}\Big(\big\{\{*\}\big\}\Big) \iff \exists B = \{*\}.\exists C_* \in \big\{\{a\},\{b\}\big\}.C = C_*$$

so

$$g^{\#}\bigg(f^{\#}\Big(\big\{\{1,2\}\big\}\Big)\bigg) = g^{\#}\Big(\big\{\{*\}\big\}\Big) = \big\{\{a\},\{b\}\big\}.$$

On the other hand, $g^{\#} \circ f : X \to \mathcal{P}\mathcal{P}Z$ is defined by:

$$g^{\#}\big(f(1)\big) = g^{\#}\big(f(2)\big) = g^{\#}\Big(\big\{\{*\}\big\}\Big) = \big\{\{a\},\{b\}\big\}.$$

So calculate from (28):

$$C \in (g^{\#} \circ f)^{\#}\Big(\big\{\{1,2\}\big\}\Big) \iff \exists A = \{1,2\}.\exists C_1, C_2 \in \big\{\{a\},\{b\}\big\}.C = C_1 \cup C_2$$

therefore

$$(g^{\#} \circ f)^{\#}\Big(\big\{\{1,2\}\big\}\Big) = \big\{\{a\},\{b\},\{a,b\}\big\}$$

hence

$$(g^{\#} f)^{\#}\Big(\big\{\{1,2\}\big\}\Big) \neq g^{\#}\bigg(f^{\#}\Big(\big\{\{1,2\}\big\}\Big)\bigg).$$

References

[1] Jon Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, pages 119–140, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.

[2] Meven Bertrand and Jurriaan Rot. Coalgebraic determinization of alternating automata. *CoRR*, abs/1804.02546, 2018.

[3] Mikołaj Bojańczyk. Recognisable languages over monads. In Igor Potapov, editor, *Developments in Language Theory*, pages 1–13. Springer International Publishing, 2015.

[4] Filippo Bonchi and Fabio Zanasi. Bialgebraic Semantics for Logic Programming. *Logical Methods in Computer Science*, 11, 2015.

[5] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, January 1981.

[6] Corina Cirstea. Canonical coalgebraic linear time logics. In Lawrence S. Moss and Pawel Sobocinski, editors, *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*, volume 35 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66–85, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[7] Fredrik Dahlqvist and Renato Neves. Program semantics as Kleisli representations. to appear.

[8] Erik P. de Vink and Jan J. M. M. Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. *Theoretical Computer Science*, 221(1-2):271–293, 1999.

[9] H. Peter Gumm and Tobias Schröder. Types and coalgebraic structure. *Algebra Universalis*, 53(2-3):229–252, 2005.

[10] Helle Hvid Hansen, Clemens Kupke, and Eric Pacuit. Neighbourhood structures: Bisimilarity and basic model theory. *Logical Methods in Computer Science*, Volume 5, Issue 2, 2009.

[11] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Log. Meth. Comp. Sci.*, 3, 2007.

[12] Paul Hudak, Simon L. Peyton Jones, Philip Wadler, Brian Boutel, Jon Fairbairn, Joseph H. Fasel, María M. Guzmán, Kevin Hammond, John Hughes, Thomas Johnsson, Richard B. Kieburtz, Rishiyur S. Nikhil, Will Partain, and John Peterson. Report on the programming language Haskell, a non-strict, purely functional language. *SIGPLAN Notices*, 27(5):1, 1992.

[13] Bart Jacobs. Trace semantics for coalgebras. *Electronic Notes in Theoretical Computer Science*, 106:167–184, 2004. Proceedings of the Workshop on Coalgebraic Methods in Computer Science (CMCS).

[14] Bart Jacobs, Paul B. Levy, and Jurriaan Rot. Steps and traces. In Corina Cirstea, editor, *14th International Workshop on Coalgebraic Methods in Computer Science, CMCS 2018*. Springer International Publishing, 2018. To appear.

[15] Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *Journal of Computer and System Sciences*, 81(5):859 – 879, 2015.

[16] Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. In *FoSSaCS 2015. Proceedings*, pages 151–166, 2015.

[17] Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. *Log. Meth. Comp. Sci.*, 12, 2016.

[18] Alexander Kurz, Stefan Milius, Dirk Pattinson, and Lutz Schröder. Simplified coalgebraic trace equivalence. In *Software, Services, and Systems: Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, pages 75–90. Springer International Publishing, 2015.

[19] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, second edition, 1998.

[20] Ernest G. Manes. Taut monads and T0-spaces. *Theor. Comput. Sci.*, 275(1-2):79–109, 2002.

[21] Ernie Manes. Monads of sets. *Handbook of algebra*, 3:67–153, 2003.

[22] Ernie Manes and Philip Mulry. Monad compositions I: general constructions and recursive distributive laws. *Theory and Applications of Categories*, 18(7):172–208, 2007.

[23] Ernie Manes and Philip Mulry. Monad compositions II: Kleisli strength. *Math. Struct. Comp. Sci.*, 18(3):613–643, 2008.

[24] Ernie G Manes. Implementing collection classes with monads. *Math. Struct. in Comp. Sci.*, 8(3):231–276, 1998.

[25] Stefan Milius, Thorsten Palm, and Daniel Schwencke. Complete iterativity for algebras with effects. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *Algebra and Coalgebra in Computer Science*, pages 34–48, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[26] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

[27] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. Artima Inc, 2008.

[28] John Power and Daniele Turi. A coalgebraic foundation for linear time semantics. *Electronic Notes in Theoretical Computer Science*, 29:259 – 274, 1999. CTCS '99, Conference on Category Theory and Computer Science.

[29] Alexandra Silva, Filippo Bonchi, Marcello Bonsangue, and Jan Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, Volume 9, Issue 1, 2013.

[30] Stanisław Szawiel and Marek Zawadowski. Monads of regular theories. *Applied Categorical Structures*, 23(3):215–262, 2015.

[31] Johan van Benthem, Nick Bezhanishvili, and Sebastian Enqvist. A propositional dynamic logic for instantial neighborhood models. In Alexandru Baltag, Jeremy Seligman, and Tomoyuki Yamada, editors, *Logic, Rationality, and Interaction*, pages 137–150, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[32] Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006.

[33] Philip Wadler. Comprehending monads. In *LISP and Functional Programming*, pages 61–78, 1990.

[34] Philip Wadler. Monads for functional programming. In *Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques-Tutorial Text*, pages 24–52, London, UK, UK, 1995. Springer-Verlag.