A Compositional Approach to Defining Logics for Coalgebras

Corina Cîrstea

School of Electronics and Computer Science
University of Southampton

Abstract

We present a compositional approach to defining expressive logics for coalgebras of endofunctors on Set. This approach uses a notion of language constructor and an associated notion of semantics to capture one inductive step in the definition of a language for coalgebras and of its semantics. We show that suitable choices for the language constructors and for their associated semantics yield logics which are both adequate and expressive w.r.t. behavioural equivalence. Moreover, we show that type-building operations give rise to corresponding operations both on language constructors and on their associated semantics, thus allowing the derivation of expressive logics for increasingly complex coalgebraic types. Our framework subsumes several existing approaches to defining logics for coalgebras, and at the same time allows the derivation of new logics, with logics for probabilistic systems being the prime example.

Key words: coalgebra, modal logic, behavioural equivalence, Hennessy-Milner property

1 Introduction

Existing modal logics for coalgebras can be classified into three categories, depending on the types of coalgebraic structures they refer to, as well as on the degree of abstraction of the modal operators they employ. The first category consists of logics which are generic in coalgebraic types, and whose associated languages are derived directly from types [11]. While both natural and expressive, these logics employ modal operators of an abstract nature, and as a

Email address: `cc2@ecs.soton.ac.uk` (Corina Cîrstea).
URL: `www.ecs.soton.ac.uk/~cc2` (Corina Cîrstea).

result are difficult to use in practice. The second category of logics concerns an inductively-defined class of coalgebraic types [13,9]. The specific nature of these types is reflected in the associated languages, which employ concrete modal operators defined inductively over types. While restrictive from the point of view of the types considered, these logics are intrinsically compositional as far as the definition of the corresponding languages and of their semantics is concerned. Finally, the third category of logics aims to combine the benefits of the previous two categories by providing reasonably concrete languages for arbitrarily general coalgebraic structures [12]. However, this is achieved at the expense of losing the naturality of the logics: rather than being determined by the coalgebraic types in question, the languages employed by these logics are based on modal operators which have to be provided explicitly. Thus, the structure of the underlying types is not usually reflected in the resulting languages. Furthermore, additional constraints on the collection of modal operators used are needed to guarantee that the resulting logics are expressive, and these constraints are not well-behaved w.r.t. type composition – it is not, in general, possible to derive expressive logics for compositions of coalgebraic types from expressive logics for the types being composed [1] .

This paper describes a compositional approach to defining expressive logics for arbitrary coalgebraic types. Our approach is based on a generalisation of the technique used in [13] to derive languages for inductively-defined endofunctors, to arbitrary endofunctors. We use an abstract notion of *language constructor* to formalise one inductive step in the definition of a modal language for coalgebras. A language constructor essentially specifies how formulae containing an extra degree of nesting of the modal operators can be constructed from an already existing set of formulae. Given a language constructor S and an endofunctor T (the latter specifying a coalgebraic type), we use a notion of T-*semantics for* S to specify how the formulae in $\mathsf{S}\mathcal{L}$ are to be interpreted over a semantic domain of form $\mathsf{T}X$, given an interpretation of the formulae in $\mathcal{L}$ over the semantic domain X. Thus, a T-semantics captures one-step in the definition of the semantics of a language for T-coalgebras. On the syntactical side, successive applications of the language constructor S eventually yield a language for T-coalgebras, whereas on the semantical side, successive applications of a T-semantics for S induce a coalgebraic semantics for this language. Moreover, it is possible to infer that the resulting language for T-coalgebras is expressive from an expressiveness condition on the T-semantics in question. Thus, in order to derive an expressive language for T-coalgebras, it suffices to exhibit a suitable language constructor and associated T-semantics. We

[1] This is simply because the class of endofunctors for which expressive logics of this kind exist is not closed under functor composition. An example here is provided by the functor $\mathcal{P}_\omega \circ \mathcal{P}_\omega$, with $\mathcal{P}_\omega : \mathsf{Set} \to \mathsf{Set}$ denoting the finite powerset functor: while expressive logics exist for both $\mathcal{P}_\omega$ and $\mathcal{P}_\omega \circ \mathcal{P}_\omega$ (see e.g. [13]), an expressive logic of the kind considered in [12] exists for $\mathcal{P}_\omega$, but not for $\mathcal{P}_\omega \circ \mathcal{P}_\omega$.

also show that standard type-building operations, including product, coproduct and functor composition, induce ways of combining language constructors on the one hand, and their associated semantics on the other, with the previously-mentioned expressiveness condition being preserved by such combinations. This makes our approach to defining expressive logics for coalgebras compositional w.r.t. coalgebraic types.

Our approach subsumes all the aforementioned approaches to defining modal logics for coalgebras, as illustrated by several examples. Moreover, the modular techniques described in the paper allow us to derive expressive logics for coalgebraic types not previously studied. The examples considered in the paper include an infinitary language for $\mathcal{P}_\kappa \circ \mathcal{P}_\kappa$-coalgebras (with $\mathcal{P}_\kappa : \mathsf{Set} \to \mathsf{Set}$ denoting the κ-bounded powerset functor, with κ a regular cardinal), as well as finitary languages for a large class of probabilistic system types, including probabilistic transition systems and probabilistic automata, all of which can be modelled coalgebraically.

The main contributions of the paper can be summarised as follows:

(1) We show that languages for coalgebras can be defined inductively using a notion of language constructor and an associated notion of semantics.
(2) We also show that the expressiveness of the resulting languages can be inferred from an expressiveness condition only involving the language constructor in question and its associated semantics.
(3) We show that operations on coalgebraic types give rise to corresponding operations both on language constructors and on their associated semantics, and that these operations preserve the previously-mentioned expressiveness condition. This allows us to derive expressive logics for inductively-defined coalgebraic types.
(4) We apply our approach to derive new logics for a number of probabilistic system types.

The present paper is an extended and improved version of [6]. A notable difference between the approach presented here and the one described in [6] is the separation between syntax and semantics when formalising one step in the definition of a language for coalgebras and of its semantics. While this separation is not essential for obtaining any of the results in this paper (and indeed, the results presented here are essentially those of [6]), the fact that syntax and semantics can be treated separately adds value to our approach by making it (a) more easily applicable to concrete coalgebraic types, and (b) more consistent with standard practice in defining logics. The separation between syntax and semantics was, to a large extent, prompted by joint work with Dirk Pattinson on the derivation of proof systems in a modular fashion [7] (see also Section 7). There, the largely syntactic nature of proof systems requires a clear separation between syntactical and semantical aspects.

The paper is structured as follows. Section 2 recalls some coalgebraic concepts required in subsequent sections, and at the same time outlines some existing approaches to defining modal logics for coalgebras. Section 3 introduces the notion of language constructor, and exemplifies it using existing logics for coalgebras. Section 4 introduces the notion of T-semantics for a language constructor, again exemplifying it with familiar logics. The expressiveness condition required to derive expressive languages for T-coalgebras is considered in Section 5. Section 6 describes a method for deriving an expressive language for T-coalgebras from a suitably-chosen T-semantics, and subsequently instantiates this method in order to derive expressive logics for coalgebras of some concrete endofunctors. The compositional nature of the approach is emphasised in each of the Sections 3, 4, 5 and 6. Section 7 discusses related work, while Section 8 summarises the results presented.

2 Preliminaries

The setting we shall be working in is that of coalgebras of endofunctors on Set. We will sometimes assume that these endofunctors are *κ-accessible* (that is, they preserve κ-filtered colimits), for some regular cardinal κ. For an accessible endofunctor T, we write $\mathsf{rank}(\mathsf{T})$ for the smallest cardinal κ for which T is κ-accessible. The endofunctors considered in the sequel will also be assumed to preserve weak pullbacks. The class of weak pullback preserving endofunctors is sufficiently general to account for most known examples of coalgebraic types. The following property of weak pullback preserving endofunctors will prove useful later in the paper.

Remark 1 *Endofunctors which preserve weak pullbacks also preserve weak limits of* W*-shaped diagrams. This follows from weak limits for such diagrams being obtained from weak pullbacks for their left and right (*V*-shaped) subdiagrams, by subsequently constructing another weak pullback.*

For an endofunctor $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$, a T-*coalgebra* is a pair $\langle C, \gamma \rangle$, with C a set (the *carrier* of the coalgebra) and $\gamma : C \to \mathsf{T}C$ a function (the *coalgebra map*). Also, a T-*coalgebra homomorphism* between T-coalgebras $\langle C, \gamma \rangle$ and $\langle D, \delta \rangle$ is a function $f : C \to D$ additionally satisfying $\mathsf{T}f \circ \gamma = \delta \circ f$. The elements of the carrier of a coalgebra are interpreted as the states of a system. Under this interpretation, the coalgebra map defines a generalised transition structure on the states, whereas coalgebra homomorphisms are required to preserve this structure. The category of T-coalgebras and T-coalgebra homomorphisms is denoted $\mathsf{Coalg}(\mathsf{T})$.

There are two standard ways of defining an observational equivalence relation between the states of (possibly different) coalgebras. The first relates two states if they can be identified by some coalgebra homomorphisms, whereas the second relates two states if they are the images of some other state of another coalgebra under some coalgebra homomorphisms. The formal definitions are as follows.

Given T-coalgebras $\langle C, \gamma \rangle$ and $\langle D, \delta \rangle$, two states $c \in C$ and $d \in D$ are *behaviourally equivalent* if there exist T-coalgebra homomorphisms $f : \langle C, \gamma \rangle \to \langle E, \eta \rangle$ and $g : \langle D, \delta \rangle \to \langle E, \eta \rangle$ with $f(c) = g(d)$. In the presence of a final T-coalgebra, behavioural equivalence coincides with equality under the unique homomorphisms into the final coalgebra (see e.g. [12, Theorem 3.4]).

A *bisimulation* between T-coalgebras $\langle C, \gamma \rangle$ and $\langle D, \delta \rangle$ is a relation $\langle R, \pi_1, \pi_2 \rangle$ on $C \times D$, with R carrying a (not necessarily unique) T-coalgebra structure $\rho : R \to \mathsf{T}R$ that makes $\pi_1 : R \to C$ and $\pi_2 : R \to D$ T-coalgebra homomorphisms. The largest bisimulation between $\langle C, \gamma \rangle$ and $\langle D, \delta \rangle$ is called *bisimilarity*. In general, bisimilarity is a stronger notion than behavioural equivalence. However, if T preserves weak pullbacks, the two notions coincide.

In constructing a final coalgebra of an endofunctor, an important rôle is played by the final sequence of the endofunctor.

Definition 2 (Final Sequence) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$. The **final sequence of** T is an ordinal-indexed sequence of sets (Z_α) together with a family $(p^\alpha_\beta)_{\beta \leq \alpha}$ of functions $p^\alpha_\beta : Z_\alpha \to Z_\beta$, satisfying:*

- $Z_{\alpha+1} = \mathsf{T}Z_\alpha$
- $p^{\alpha+1}_{\beta+1} = \mathsf{T}p^\alpha_\beta$ *for* $\beta \leq \alpha$
- $p^\alpha_\alpha = 1_{Z_\alpha}$
- $p^\alpha_\gamma = p^\beta_\gamma \circ p^\alpha_\beta$ *for* $\gamma \leq \beta \leq \alpha$
- *if α is a limit ordinal, the cone $Z_\alpha, (p^\alpha_\beta)_{\beta<\alpha}$ for $(p^\beta_\gamma)_{\gamma \leq \beta < \alpha}$ is limiting.*

The final sequence of T is uniquely defined by the above conditions. In particular, $Z_0 = 1$, with $1 = \{0\}$ denoting a final object in Set.

If the final sequence of T *stabilises at* α (that is, if $p^{\alpha+1}_\alpha$ is an isomorphism) for some ordinal α, then Z_α is the carrier of a final T-coalgebra (see [2, Theorem 1.3], or [1, Theorem 5]). Various constraints on T can be used to ensure that its final sequence stabilises at a specific α. In particular, if T is ω^{op}-continuous, its final sequence stabilises at ω. Also, if T is κ-accessible, with κ a regular cardinal, its final sequence stabilises at $\kappa \cdot 2$ (see [16, Theorem 10]).

Remark 3 *For a T-coalgebra $\langle C, \gamma \rangle$, one can define an ordinal-indexed sequence of functions (γ_α), with $\gamma_\alpha : C \to Z_\alpha$, as follows:*

- $\gamma_\alpha = \mathsf{T}\gamma_\beta \circ \gamma$, *if* $\alpha = \beta + 1$;

- γ_α *is the unique function satisfying* $p^\alpha_\beta \circ \gamma_\alpha = \gamma_\beta$ *for each* $\beta < \alpha$, *if* α *is a limit ordinal.*

The functions γ_α *define a cone over the final sequence of* T. *Moreover, it can be shown by transfinite induction on* α *that* $\delta_\alpha \circ f = \gamma_\alpha$ *whenever* $f : \langle C, \gamma\rangle \to \langle D, \delta\rangle$ *is a* T*-coalgebra homomorphism; that is,* $f : C \to D$ *defines a morphism of cones from* (γ_α) *to* (δ_α).

Remark 4 *It is shown in [12, Theorem 3.4] that if* T *is* κ*-accessible and* $\langle C, \gamma\rangle$ *and* $\langle D, \delta\rangle$ *are* T*-coalgebras, then the relation on* $C \times D$ *defined by* $c \sim d$ *if* $\gamma_\kappa(c) = \delta_\kappa(d)$ *for* $\langle c, d\rangle \in C \times D$ *coincides with behavioural equivalence.*

In what follows, we will be interested in languages able to formalise properties of states of coalgebras up to behavioural equivalence.

Definition 5 (Adequacy, Expressiveness) *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$, *let* $\mathcal{L}$ *be a set of formulae, and let* $\models \; = (\models_\gamma)_{\langle C,\gamma\rangle \in |\mathsf{Coalg}(\mathsf{T})|}$ *be a* $|\mathsf{Coalg}(\mathsf{T})|$*-indexed family of relations, with* $\models_\gamma$ *a relation on* $C \times \mathcal{L}$ *for any* $\langle C, \gamma\rangle \in |\mathsf{Coalg}(\mathsf{T})|$. *Given* T*-coalgebras* $\langle C, \gamma\rangle$ *and* $\langle D, \delta\rangle$, *two states* $c \in C$ *and* $d \in D$ *are* ***logically equivalent*** *if* $c \models_\gamma \varphi$ *precisely when* $d \models_\delta \varphi$, *for any* $\varphi \in \mathcal{L}$. *The language defined by* $\mathcal{L}$ *and* $\models$ *is* ***adequate*** *if behavioural equivalence implies logical equivalence, and* ***expressive*** *if logical equivalence implies behavioural equivalence (for any* T*-coalgebras* $\langle C, \gamma\rangle$ *and* $\langle D, \delta\rangle$ *and any states* $c \in C$ *and* $d \in D$*). Also, the language* ***captures behavioural equivalence*** *if it is both adequate and expressive.*

The observation in Remark 4 will be exploited later in the paper, in order to obtain a language for T-coalgebras which captures behavioural equivalence. We note that, in the case of weak pullback preserving endofunctors, such a language also captures bisimulation.

We now recall several existing approaches to defining modal logics for coalgebras of endofunctors on Set. We begin with the coalgebraic logic defined by Moss [11].

Definition 6 (Coalgebraic Logic) *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *be an accessible, weak pullback preserving and inclusion preserving endofunctor. The* ***language*** $\mathcal{L}_\mathsf{T}$ ***of coalgebraic logic*** *is the carrier of the initial algebra of the functor* $X \mapsto \mathcal{P}X + \mathsf{T}X$. *The injections arising from* $\mathcal{P}\mathcal{L}_\mathsf{T} + \mathsf{T}\mathcal{L}_\mathsf{T} \simeq \mathcal{L}_\mathsf{T}$ *are denoted* $\bigwedge : \mathcal{P}\mathcal{L}_\mathsf{T} \to \mathcal{L}_\mathsf{T}$ *and* $\nabla : \mathsf{T}\mathcal{L}_\mathsf{T} \to \mathcal{L}_\mathsf{T}$.

For a T*-coalgebra* $\langle C, \gamma\rangle$, *the* ***satisfaction relation*** $\models_\gamma$ *between elements of* C *and formulae of* $\mathcal{L}_\mathsf{T}$ *is defined inductively as follows:*

$$\begin{array}{rcl} c \models_\gamma \bigwedge \Phi & \textit{iff} & c \models_\gamma \varphi \ \textit{ for all } \ \varphi \in \Phi \\ c \models_\gamma \nabla \psi & \textit{iff} & \gamma(c)\,(\mathsf{T}{\models_\gamma})\,\psi \end{array}$$

for $c \in C$, $\Phi \in \mathcal{P}\mathcal{L}_{\mathsf{T}}$ and $\psi \in \mathsf{T}\mathcal{L}_{\mathsf{T}}$, where $\mathsf{T}\!\models_\gamma \subseteq \mathsf{T}C \times \mathsf{T}\mathcal{L}_{\mathsf{T}}$ denotes the image of the relation $\models_\gamma \subseteq C \times \mathcal{L}_{\mathsf{T}}$ under the lifting of T to relations (see [11] for details).

The language $\mathcal{L}_{\mathsf{T}}$ of coalgebraic logic is both adequate and expressive (see [11] or [12, Section 5]).

We now move to a more specific class of endofunctors on Set, namely that of *Kripke polynomial functors*, as considered by Rößiger [13] and subsequently by Jacobs [9]. The class of Kripke polynomial functors contains the constant functor $X \mapsto A$ with A a non-empty set, as well as the identity functor $X \mapsto X$, and is closed under binary products $\mathsf{T}_1 \times \mathsf{T}_2$, binary coproducts $\mathsf{T}_1 + \mathsf{T}_2$, exponents T^A with A a finite, non-empty set, and powersets $\mathcal{P}\circ\mathsf{T}$. By replacing the powerset functor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ with the κ-bounded powerset functor $\mathcal{P}_\kappa : \mathsf{Set} \to \mathsf{Set}$ [2], with κ a regular cardinal, we obtain a variant of the notion of Kripke polynomial functor which we call *κ-bounded Kripke polynomial functor*. Then, ω-bounded Kripke polynomial functors are the *finite Kripke polynomial functors* of [13,9]. All Kripke polynomial functors preserve weak pullbacks. Moreover, κ-bounded Kripke polynomial functors are κ-accessible.

Kripke polynomial functors can be used to model *deterministic systems* (via the functor $X \mapsto X^A$), *non-deterministic systems* (via the functor $X \mapsto (\mathcal{P}X)^A$), *deterministic automata* (via the functor $X \mapsto \{0,1\} \times X^A$), as well as other similar kinds of systems.

A many-sorted modal logic for Kripke polynomial functors was defined in [9] (see also [13]). The formulae of this logic are defined inductively on the structure of Kripke polynomial functors T, with each ingredient functor used in the definition of T giving a rise to a sort for formulae. The semantics of the resulting modal language is itself defined by induction (on the structure of T as well as on the structure of formulae), see [9] for details.

The modal logic of [9] does not distinguish any bisimilar states (see [9, Corollary 3.7], and therefore it is adequate [3]. However, the logic is only expressive in the case of *finite* Kripke polynomial functors (see [9, Corollary 5.9]).

Among the endofunctors which are not covered by the approach in [13,9], but which present an interest from a practical point of view, is the *finite probability*

[2] $\mathcal{P}_\kappa$ takes a set X to the set of subsets of X of cardinality smaller than κ.

[3] Since Kripke polynomial functors preserve weak pullbacks, the induced notions of behavioural equivalence and bisimilarity coincide.

distribution functor $\mathcal{D}_\omega : \mathsf{Set} \to \mathsf{Set}$, defined by:

$$\mathcal{D}_\omega X = \{ \ \mu : X \to [0,1] \mid \mu(x) \neq 0 \ \text{ for finitely many } x \in X \ , \\ \sum_{x \in X} \mu(x) = 1 \ \} \ \text{ for } X \in |\mathsf{Set}|$$

$$(\mathcal{D}_\omega f)(\mu)(y) = \sum_{f(x)=y} \mu(x) \ \text{ for } f : X \to Y \ , \ \mu \in \mathcal{D}_\omega X \ , \text{ and } y \in Y$$

The endofunctor $\mathcal{D}_\omega$ preserves weak pullbacks (see [11]). Also, one can easily show that $\mathcal{D}_\omega$ is ω-accessible.

Endofunctors similar to Kripke polynomial functors but which also incorporate the finite probability distribution functor in their definition are typically used to model probabilistic systems (see [3] for an overview). However, the only such types for which suitable logics have been investigated correspond to the functors $1 + \mathcal{D}_\omega$ and respectively $(1 + \mathcal{D}_\omega)^A$ (see [10]). Coalgebras of these functors are known as *(labelled) probabilistic transition systems*, and have been studied by de Vink and Rutten [8], where it was shown that the standard notion of probabilistic bisimulation, as defined in [10], coincides with coalgebraic bisimulation.

The last approach to defining logics for coalgebras which we outline here is due to Pattinson [12] and aims to cover arbitrary endofunctors while retaining the simplicity of the resulting languages.

Definition 7 (Logic Induced by Predicate Liftings) *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$. *A* ***predicate lifting for*** T *is a natural transformation* $\lambda : \hat{\mathcal{P}} \Rightarrow \hat{\mathcal{P}} \circ \mathsf{T}$ *(with* $\hat{\mathcal{P}} : \mathsf{Set} \to \mathsf{Set}$ *denoting the contravariant powerset functor). A set* Λ *of predicate liftings is* ***separating*** *if the map* $t \in \mathsf{T}X \mapsto \{ \lambda_X(Y) \mid \lambda \in \Lambda\, , Y \in \mathcal{P}X\, , \lambda_X(Y) \ni t \}$ *is monic for any set* X.

Given a regular cardinal κ *and a set* Λ *of predicate liftings, the* ***language*** $\mathcal{L}_\kappa(\Lambda)$ ***induced by*** κ ***and*** Λ *is defined inductively by:*

$$\varphi ::= \bigwedge \Phi \mid \neg\varphi \mid [\lambda]\varphi\, , \quad \Phi \in \mathcal{P}_\kappa(\mathcal{L}_\kappa(\Lambda))\, , \quad \varphi \in \mathcal{L}_\kappa(\Lambda)\, , \quad \lambda \in \Lambda$$

In addition, one defines $\langle\lambda\rangle\varphi ::= \neg[\lambda]\neg\varphi$ *for* $\lambda \in \Lambda$ *and* $\varphi \in \mathcal{L}_\kappa(\Lambda)$.

For a T*-coalgebra* $\langle C, \gamma\rangle$*, the* ***satisfaction relation*** $\models_\gamma$ *between elements of* C *and formulae of* $\mathcal{L}_\kappa(\Lambda)$ *is defined inductively by:*

$$\begin{aligned} c \models_\gamma \bigwedge \Phi \ &\textit{ iff } \ c \models_\gamma \varphi \ \textit{ for all } \ \varphi \in \Phi \\ c \models_\gamma \neg\varphi \ &\textit{ iff } \ c \not\models_\gamma \varphi \\ c \models_\gamma [\lambda]\varphi \ &\textit{ iff } \ \gamma_C(c) \in \lambda_C([\![\varphi]\!]_\gamma) \end{aligned}$$

where $[\![\varphi]\!]_\gamma ::= \{c \in C \mid c \models_\gamma \varphi\}$.

It is shown in [12] that the language $\mathcal{L}_\kappa(\Lambda)$ is adequate; and if, in addition, T is accessible and Λ is separating, then there exists a cardinal κ, which depends only on $\mathsf{rank}(\mathsf{T})$ and $\mathsf{card}(\Lambda)$, such that $\mathcal{L}_\kappa(\Lambda)$ is expressive.

3 Language Constructors, Compositionally

We now fix an arbitrary regular cardinal κ. In the sequel, we shall consider languages which are closed under conjunctions of cardinality smaller than κ, as well as under negation. Such languages will be regarded as algebras of the functor $\mathsf{B}_\kappa = \mathcal{P}_\kappa + \mathsf{Id} : \mathsf{Set} \to \mathsf{Set}$, with the two components of the algebra maps taking sets of formulae Φ of cardinality smaller than κ to their conjunction $\bigwedge \Phi$, and respectively single formulae φ to their negation $\neg\varphi$. We also write $\top$ for $\bigwedge \emptyset$, $\bot$ for $\neg\top$, and $\bigvee \Phi$ for $\neg \bigwedge_{\varphi\in\Phi} \neg\varphi$. Later in the paper, we shall instantiate κ to suit our purposes.

The notion of language constructor which we now introduce aims to capture *one inductive step* in the definition of a language for T-coalgebras.

Definition 8 (Language Constructor) *A **language constructor** is a functor* $\mathsf{S} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$.

We now give some examples of language constructors, based on the logics outlined in Section 2.

Example 9 (Language Constructor for Coalgebraic Logic) *Suppose* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *is as in Definition 6, and define* $\mathsf{S}_\mathsf{T} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *by:*

- *for a* B_κ*-algebra* $\mathcal{L}$*,* $\mathsf{S}_\mathsf{T}\mathcal{L}$ *is the free* B_κ*-algebra* $(\mathsf{T}\mathcal{L})^{\wedge,\neg}$ *over* $\mathsf{T}\mathcal{L}$*;*
- *for a* B_κ*-algebra homomorphism* $l : \mathcal{L} \to \mathcal{L}'$*,* $\mathsf{S}_\mathsf{T} l$ *is the* B_κ*-algebra homomorphism* $(\mathsf{T}l)^{\wedge,\neg} : (\mathsf{T}\mathcal{L})^{\wedge,\neg} \to (\mathsf{T}\mathcal{L}')^{\wedge,\neg}$

where $(\_)^{\wedge,\neg} : \mathsf{Set} \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *denotes the left adjoint to the functor taking* B_κ*-algebras to their carrier. The language constructor* S_T *mirrors the construction of the language of coalgebraic logic [11]. The difference w.r.t. [11] is that here the size of the conjunctions is bounded by* κ*, and moreover, negation is also present.*

Example 10 (Inductively-Defined Language Constructors) *Consider the class of endofunctors generated by the following syntax:*

$$\mathsf{T} ::= A \mid \mathsf{Id} \mid \mathcal{P}_\kappa \mid \mathcal{D}_\omega \mid \mathsf{T}_1 + \mathsf{T}_2 \mid \mathsf{T}_1 \times \mathsf{T}_2 \mid (\mathsf{T}_1)^A \mid \mathsf{T}_1 \circ \mathsf{T}_2$$

This class includes all Kripke polynomial functors; in addition, it contains the finite probability distribution functor, and it is closed under functor com-

position. To each endofunctor T *of the above form, we associate a language constructor* $\mathsf{S_T}$ *as follows:*

- *If* A *is a non-empty set,* $\mathsf{S}_A : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *is the constant functor taking* $\mathcal{L}$ *to* $A^{\wedge,\neg}$.
- $\mathsf{S_{Id}} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *is the identity functor on* $\mathsf{Alg}(\mathsf{B}_\kappa)$.
- $\mathsf{S}_{\mathcal{P}_\kappa} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *takes* $\mathcal{L}$ *to* $\{\, \Box\varphi \mid \varphi \in \mathcal{L}\,\}^{\wedge,\neg}$.
  We write $\Diamond\varphi$ *for* $\neg\Box(\neg\varphi) \in \{\, \Box\varphi \mid \varphi \in \mathcal{L}\,\}^{\wedge,\neg}$, *for* $\varphi \in \mathcal{L}$.
- $\mathsf{S}_{\mathcal{D}_\omega} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *takes* $\mathcal{L}$ *to* $\{\, \Diamond_p\varphi \mid p \in [0,1]\,,\ \varphi \in \mathcal{L}\,\}^{\wedge,\neg}$.
- *If* $\mathsf{S}_{\mathsf{T}_i} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *for* $i \in \{1,2\}$ *and* A *is a non-empty set, define:*
  · $(\mathsf{S}_{\mathsf{T}_1+\mathsf{T}_2})\mathcal{L} ::= \mathsf{S}_{\mathsf{T}_1}\mathcal{L} \oplus \mathsf{S}_{\mathsf{T}_2}\mathcal{L}$
  · $(\mathsf{S}_{\mathsf{T}_1\times\mathsf{T}_2})\mathcal{L} ::= \mathsf{S}_{\mathsf{T}_1}\mathcal{L} \otimes \mathsf{S}_{\mathsf{T}_2}\mathcal{L}$
  · $(\mathsf{S}_{(\mathsf{T}_1)^A})\mathcal{L} ::= (\mathsf{S}_{\mathsf{T}_1}\mathcal{L})^A$
  · $(\mathsf{S}_{\mathsf{T}_1\circ\mathsf{T}_2})\mathcal{L} ::= \mathsf{S}_{\mathsf{T}_1}\mathsf{S}_{\mathsf{T}_2}\mathcal{L}$
  where the binary operations $\oplus\,,\ \otimes : \mathsf{Alg}(\mathsf{B}_\kappa) \times \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *and the unary operation* $(\_)^A : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *are defined by:*
  · $\mathcal{L}_1 \oplus \mathcal{L}_2 = (\mathcal{L}_1 + \mathcal{L}_2)^{\wedge,\neg}$
  · $\mathcal{L}_1 \otimes \mathcal{L}_2 = (\mathcal{L}_1 \times \mathcal{L}_2)^{\wedge,\neg}$
  · $(\mathcal{L}_1)^A = (\prod_{a\in A} \mathcal{L}_1)^{\wedge,\neg}$.
  We note that, in the definitions of $\mathcal{L}_1 \oplus \mathcal{L}_2$, $\mathcal{L}_1 \otimes \mathcal{L}_2$ *and* $(\mathcal{L}_1)^A$, *the products and the coproduct are computed in* Set, *while the subsequent closure under* $\wedge$ *and* $\neg$ *ensures that the expression in question yields a* B_κ*-algebra. If* $\varphi_i \in \mathcal{L}_i$ *for* $i = 1, 2$, *we define:*

  $$\langle\kappa_i\rangle\varphi_i ::= \iota_i(\varphi_i) \in \mathcal{L}_1 + \mathcal{L}_2 \qquad [\kappa_i]\varphi_i ::= \neg\langle\kappa_i\rangle\neg\varphi_i \in (\mathcal{L}_1 + \mathcal{L}_2)^{\wedge,\neg}$$
  $$[\pi_1]\varphi_1 ::= \langle\varphi_1, \top\rangle \in \mathcal{L}_1 \times \mathcal{L}_2 \qquad [\pi_2]\varphi_2 ::= \langle\top, \varphi_2\rangle \in \mathcal{L}_1 \times \mathcal{L}_2$$
  $$[a]\varphi_1 ::= (\varphi_b)_{b\in A} \in \prod_{a\in A} \mathcal{L}_1 \quad \textit{with} \quad \varphi_b = \begin{cases} \varphi_1, & \textit{if} \ \ b = a \\ \top, & \textit{if} \ \ b \neq a \end{cases}$$

  This notation makes explicit the link between the modal operators used here and the ones of [9].

Each language constructor $\mathsf{S_T}$ *defined above specifies one step in the definition of a language for* T*-coalgebras. For instance, in the case of the* κ*-bounded powerset functor, this step involves closing under a unary modal operator* $\Box$, *and subsequently closing under* $\wedge$ *and* $\neg$. *Similarly, a collection of unary modal operators* $\Diamond_p$ *indexed by probability values is used in the case of the finite probability distribution functor. When combining language constructors, the intention is to capture one step in the definition of a language for* $\mathsf{T}_1 + \mathsf{T}_2$*-,* $\mathsf{T}_1 \times \mathsf{T}_2$*- and* $\mathsf{T}_1 \circ \mathsf{T}_2$*-coalgebras, respectively, assuming that the language constructors we are combining capture one step in the definition of a language for* T_i*-coalgebras.*

Example 11 (Language Constructors from Predicate Liftings) *Sets of predicate liftings also induce language constructors. For an endofunctor* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *and a set* Λ *of predicate liftings for* T*, define* $\mathsf{S}_\Lambda : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *by:*

- $\mathsf{S}_\Lambda \mathcal{L} = \{ [\lambda]\varphi \mid \lambda \in \Lambda,\ \varphi \in \mathcal{L} \}^{\wedge,\neg}$ *for each* B_κ*-algebra* $\mathcal{L}$ *;*
- $(\mathsf{S}_\Lambda l)([\lambda]\varphi) = [\lambda]l(\varphi)$ *for each* B_κ*-algebra homomorphism* $l : \mathcal{L} \to \mathcal{L}'$, $\lambda \in \Lambda$ *and* $\varphi \in \mathcal{L}$.

4 Language Semantics, Compositionally

We now move on to defining a semantics for a language constructor S. Such a semantics also depends on an endofunctor T, and specifies how formulae in $\mathsf{S}\mathcal{L}$ are to be interpreted over the set $\mathsf{T}X$, given an interpretation of formulae in $\mathcal{L}$ over the set X. Thus, a T-*semantics for* S captures one step in the definition of a semantics of a language for T-coalgebras.

We use *comma categories* (see e.g. [4, Section 1.6]) to define interpretations of sets of formulae over particular semantic domains. Specifically, we consider the comma category $(\mathsf{Id}, \check{\mathcal{P}})$, with $\mathsf{Id} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ denoting the identity functor, and with $\check{\mathcal{P}} : \mathsf{Set}^{\mathrm{op}} \to \mathsf{Alg}(\mathsf{B}_\kappa)$ taking a set X to the B_κ-algebra $(\mathcal{P}X, \bigcap, ^-)$ [4], and a function $f : X' \to X$ to the B_κ-algebra homomorphism $\hat{\mathcal{P}}f : \mathcal{P}X \to \mathcal{P}X'$.

Definition 12 (Interpreted Language) *An* ***interpreted language*** *is an object of the category* $(\mathsf{Id}, \check{\mathcal{P}})$*, while a* ***map between interpreted languages*** *is an arrow of this category. The category* $(\mathsf{Id}, \check{\mathcal{P}})$ *will henceforth be denoted* $\mathsf{IntLang}$.

An interpreted language is therefore given by a tuple $\langle \mathcal{L}, X, d \rangle$, with $\mathcal{L}$ a B_κ-algebra, X a set, and $d : \mathcal{L} \to \mathcal{P}X$ a B_κ-algebra homomorphism. (We will occasionally refer to an interpreted language $\langle \mathcal{L}, X, d \rangle$ simply via its *denotation map* $d : \mathcal{L} \to \mathcal{P}X$.) Also, a map between interpreted languages $\langle \mathcal{L}, X, d \rangle$ and $\langle \mathcal{L}', X', d' \rangle$ is given by a pair $\langle l, f \rangle$, with $l : \mathcal{L} \to \mathcal{L}'$ a B_κ-algebra homomorphism and $f : X' \to X$ a function, such that $\hat{\mathcal{P}}f \circ d = d' \circ l$ [5]:

$$\begin{tikzcd} \mathcal{L} \arrow[r, "l"] \arrow[d, "d"'] & \mathcal{L}' \arrow[d, "d'"] \\ \mathcal{P}X \arrow[r, "\hat{\mathcal{P}}f"'] & \mathcal{P}X' \end{tikzcd}$$

[4] Here, $\bigcap$ denotes set intersection whereas $^-$ denotes set complement (w.r.t. X).
[5] The commutativity of this diagram in Set implies its commutativity in $\mathsf{Alg}(\mathsf{B}_\kappa)$.

Thus, a map between interpreted languages $\langle \mathcal{L}, X, d\rangle$ and $\langle \mathcal{L}', X', d'\rangle$ defines a denotation-preserving translation between the sets of formulae $\mathcal{L}$ and $\mathcal{L}'$.

Remark 13 *Both the category* $\mathsf{IntLang}$ *and the notion of interpreted language depend on the cardinal* κ*. For simplicity of notation, we choose not to make this dependency explicit. However, this dependency will play an important rôle in Section 6, when considering the size of the conjunctions needed to obtain an expressive language for* T*-coalgebras.*

Definition 14 *Given an interpreted language* $\langle \mathcal{L}, X, d\rangle$*, we write* $x \models_d \varphi$ *for* $x \in d(\varphi)$*, with* $x \in X$ *and* $\varphi \in \mathcal{L}$*. Two elements* $x, y \in X$ *are said to be* ***logically equivalent*** *if* $x \models_d \varphi$ *precisely when* $y \models_d \varphi$*, for any* $\varphi \in \mathcal{L}$*.*

We now let $\mathsf{L} : \mathsf{IntLang} \to \mathsf{Alg}(\mathsf{B}_\kappa)$ and $\mathsf{E} : \mathsf{IntLang} \to \mathsf{Set}^{\mathrm{op}}$ denote the functors taking $\langle \mathcal{L}, X, d\rangle$ to $\mathcal{L}$ and X, respectively, and $\langle l, f\rangle : \langle \mathcal{L}, X, d\rangle \to \langle \mathcal{L}', X', d'\rangle$ to $l : \mathcal{L} \to \mathcal{L}'$ and $f : X' \to X$, respectively.

Later in the paper, we will use colimits in the category $\mathsf{IntLang}$ to combine languages interpreted over different semantic domains. The next result is a first step towards proving that colimits exist in $\mathsf{IntLang}$.

Proposition 15 *The functor* E *is a cofibration*[6] *.*

PROOF (Sketch). For an interpreted language $\langle \mathcal{L}, X, d\rangle$ and a function $f : X' \to X$, the map $\langle 1_{\mathcal{L}}, f\rangle : \langle \mathcal{L}, X, d\rangle \to \langle \mathcal{L}, X', \hat{\mathcal{P}}f \circ d\rangle$ is cocartesian.

For a set X, we write $\mathsf{IntLang}_X$ for the slice category $\mathsf{Alg}(\mathsf{B}_\kappa)/(\mathcal{P}X, \bigcap, ^-)$. The objects of $\mathsf{IntLang}_X$ are therefore pairs $\langle \mathcal{L}, d\rangle$ with $d : \mathcal{L} \to \mathcal{P}X$, while arrows from $\langle \mathcal{L}_1, d_1\rangle$ to $\langle \mathcal{L}_2, d_2\rangle$ are given by B_κ-algebra homomorphisms $l : \mathcal{L}_1 \to \mathcal{L}_2$ satisfying $d_2 \circ l = d_1$. The fact that E is a cofibration allows us to translate languages interpreted over X to languages interpreted over X', along functions $f : X' \to X$. Specifically, any such function induces a **coreindexing functor** $f_* : \mathsf{IntLang}_X \to \mathsf{IntLang}_{X'}$, which takes $\langle \mathcal{L}, d\rangle$ to $\langle \mathcal{L}, \hat{\mathcal{P}}f \circ d\rangle$, and $l : \langle \mathcal{L}_1, d_1\rangle \to \langle \mathcal{L}_2, d_2\rangle$ to $l : \langle \mathcal{L}_1, \hat{\mathcal{P}}f \circ d_1\rangle \to \langle \mathcal{L}_2, \hat{\mathcal{P}}f \circ d_2\rangle$.

We now use Proposition 15 to prove the existence of colimits in the category $\mathsf{IntLang}$, and the preservation of colimits by the functor E.

Proposition 16 $\mathsf{IntLang}$ *has colimits, and* E *preserve colimits.*

PROOF (Sketch). According to [5, Proposition 8.5.2], it suffices to verify that:

[6] See [5] for a definition of this notion.

(1) $\mathsf{Set}^{\mathrm{op}}$ has all colimits;

(2) $\mathsf{IntLang}_X$ has all colimits, for any set X;

(3) the coreindexing functor $f_* : \mathsf{IntLang}_X \to \mathsf{IntLang}_{X'}$ preserves all colimits, for any function $f : X' \to X$.

The second requirement follows immediately from the definition of $\mathsf{IntLang}_X$ and from the existence of colimits in $\mathsf{Alg}(\mathsf{B}_\kappa)$. The third requirement is a direct consequence of the way colimits are constructed in $\mathsf{IntLang}_X$ and $\mathsf{IntLang}_{X'}$ (namely from colimits in $\mathsf{Alg}(\mathsf{B}_\kappa)$).

Thus, colimits in $\mathsf{IntLang}$ are constructed from colimits in $\mathsf{Set}^{\mathrm{op}}$ and colimits in the categories $\mathsf{IntLang}_X$, using the coreindexing functors. In particular, an initial object in $\mathsf{IntLang}$ is given by $\langle \mathcal{L}_0, 1, d_0 \rangle$, with $\mathcal{L}_0$ containing $\top$ and $\bot$, and with d_0 mapping $\top$ to 1 and $\bot$ to $\emptyset$.

We now define the notion of a T-semantics for a language constructor mentioned earlier. This notion captures one step in the definition of a semantics of a language for T-coalgebras.

Definition 17 (T-Semantics for S) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ be an endofunctor, and let $\mathsf{S} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ be a language constructor. A* T***-semantics for*** S *is a functor $\mathcal{F} : \mathsf{IntLang} \to \mathsf{IntLang}$ such that $\mathsf{L} \circ \mathcal{F} = \mathsf{S} \circ \mathsf{L}$ and $\mathsf{E} \circ \mathcal{F} = \mathsf{T}^{\mathrm{op}} \circ \mathsf{E}$:*

$$\begin{tikzcd}
\mathsf{Alg}(\mathsf{B}_\kappa) \arrow[r, "\mathsf{S}"] & \mathsf{Alg}(\mathsf{B}_\kappa) \\
\mathsf{IntLang} \arrow[u, "\mathsf{L}"] \arrow[r, "\mathcal{F}"] \arrow[d, "\mathsf{E}"'] & \mathsf{IntLang} \arrow[u, "\mathsf{L}"'] \arrow[d, "\mathsf{E}"] \\
\mathsf{Set}^{\mathrm{op}} \arrow[r, "\mathsf{T}^{\mathrm{op}}"'] & \mathsf{Set}^{\mathrm{op}}
\end{tikzcd}$$

Thus, a T-semantics for S takes an interpreted language $\langle \mathcal{L}, X, d \rangle$ to an interpreted language of form $\langle \mathsf{S}\mathcal{L}, \mathsf{T}X, d' \rangle$.

Some examples of T-semantics for the language constructors defined in Section 3 are given next. We begin with the language constructor for coalgebraic logic.

Example 18 (T-Semantics for S_T) *If $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ is as in Definition 6 and $\mathsf{S}_\mathsf{T} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ is as in Example 9, then a T-semantics for S_T is given by the functor $\mathcal{F}_\mathsf{T} : \mathsf{IntLang} \to \mathsf{IntLang}$ which takes $\langle \mathcal{L}, X, d \rangle$ to $\langle \mathsf{S}_\mathsf{T}\mathcal{L}, \mathsf{T}X, (\epsilon_X \circ \mathsf{T}d)^\# \rangle$:*

$$\begin{tikzcd}
\mathcal{L} \arrow[d, "d"] & \mathsf{T}\mathcal{L} \arrow[d, "\mathsf{T}d"] & \\
\mathcal{P}X & \mathsf{T}\mathcal{P}X \arrow[r, "\epsilon_X"] & \mathcal{P}\mathsf{T}X
\end{tikzcd}$$

where the natural transformation $\epsilon : \mathsf{T} \circ \hat{\mathcal{P}} \Rightarrow \hat{\mathcal{P}} \circ \mathsf{T}$ *is given by:*

$$\epsilon_X(Y) = \{\, t \in \mathsf{T}X \mid t\ (\mathsf{T}{\in})\ Y \,\} \text{ for } X \in |\mathsf{Set}| \text{ and } Y \in \mathsf{T}\mathcal{P}X. \tag{1}$$

and where $(\epsilon_X \circ \mathsf{T}d)^{\#} : (\mathsf{T}\mathcal{L})^{\wedge,\neg} \to \mathcal{P}\mathsf{T}X$ *denotes the unique extension of* $\epsilon_X \circ \mathsf{T}d : \mathsf{T}\mathcal{L} \to \mathcal{P}\mathsf{T}X$ *to a* B_κ*-algebra homomorphism. To see that* ϵ *is indeed natural, let* $f : C \to D$ *be a function. The naturality of* ϵ *w.r.t.* f *amounts to:*

$$(\mathsf{T}f)(t)\ (\mathsf{T}{\in})\ Y \quad \textit{iff} \quad t\ (\mathsf{T}{\in})\ (\mathsf{T}\hat{\mathcal{P}}f)(Y)$$

for any $t \in \mathsf{T}C$ *and* $Y \in \mathsf{T}\mathcal{P}D$*. To prove this, note that any (weakly) limiting cone for the left diagram below gives a (weakly) limiting cone for the right diagram, and conversely:*

$$\begin{tikzcd}
C \arrow[d, "f"'] & & \mathcal{P}D \arrow[d, "1_{\mathcal{P}D}"] \\
D & \in \arrow[l] \arrow[r] & \mathcal{P}D
\end{tikzcd}
\qquad
\begin{tikzcd}
C \arrow[d, "1_C"'] & & \mathcal{P}D \arrow[d, "\hat{\mathcal{P}}f"] \\
C & \in \arrow[l] \arrow[r] & \mathcal{P}C
\end{tikzcd}$$

Then, as T *preserves weak limits of* W*-shaped diagrams (by Remark 1), the following two diagrams enjoy a similar property:*

$$\begin{tikzcd}
\mathsf{T}C \arrow[d, "\mathsf{T}f"'] & & \mathsf{T}\mathcal{P}D \arrow[d, "1_{\mathsf{T}\mathcal{P}D}"] \\
\mathsf{T}D & \mathsf{T}{\in} \arrow[l] \arrow[r] & \mathsf{T}\mathcal{P}D
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{T}C \arrow[d, "1_{\mathsf{T}C}"'] & & \mathsf{T}\mathcal{P}D \arrow[d, "\mathsf{T}\hat{\mathcal{P}}f"] \\
\mathsf{T}C & \mathsf{T}{\in} \arrow[l] \arrow[r] & \mathsf{T}\mathcal{P}C
\end{tikzcd}$$

and therefore define the same relation on $\mathsf{T}C \times \mathsf{T}\mathcal{P}D$*. But this is precisely the naturality of* ϵ *w.r.t.* f*.*

The functor $\mathcal{F}_\mathsf{T}$ *takes a map* $\langle l, f\rangle : \langle \mathcal{L}_1, X_1, d_1\rangle \to \langle \mathcal{L}_2, X_2, d_2\rangle$ *to the map* $\langle \mathsf{S}_\mathsf{T} l, \mathsf{T}f\rangle : \langle \mathsf{S}_\mathsf{T}\mathcal{L}_1, \mathsf{T}X_1, (\epsilon_{X_1} \circ \mathsf{T}d_1)^{\#}\rangle \to \langle \mathsf{S}_\mathsf{T}\mathcal{L}_2, \mathsf{T}X_2, (\epsilon_{X_2} \circ \mathsf{T}d_2)^{\#}\rangle$*:*

$$\begin{tikzcd}
\mathcal{L}_1 \arrow[r, "l"] \arrow[d, "d_1"'] & \mathcal{L}_2 \arrow[d, "d_2"] \\
\hat{\mathcal{P}}X_1 \arrow[r, "\hat{\mathcal{P}}f"] & \hat{\mathcal{P}}X_2
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{T}\mathcal{L}_1 \arrow[r, "\mathsf{T}l"] \arrow[d, "\mathsf{T}d_1"'] & \mathsf{T}\mathcal{L}_2 \arrow[d, "\mathsf{T}d_2"] \\
\mathsf{T}\hat{\mathcal{P}}X_1 \arrow[r, "\mathsf{T}\hat{\mathcal{P}}f"] \arrow[d, "\epsilon_{X_1}"'] & \mathsf{T}\hat{\mathcal{P}}X_2 \arrow[d, "\epsilon_{X_2}"] \\
\hat{\mathcal{P}}\mathsf{T}X_1 \arrow[r, "\hat{\mathcal{P}}\mathsf{T}f"] & \hat{\mathcal{P}}\mathsf{T}X_2
\end{tikzcd}$$

We now show that, if $\mathcal{F}_\mathsf{T}$ *takes* $\langle \mathcal{L}, X, d\rangle$ *to* $\langle \mathsf{S}_\mathsf{T}\mathcal{L}, \mathsf{T}X, d'\rangle$*, then the relation* $\models_{d'}$ *is given by the natural extension* $(\mathsf{T}{\models_d})^{\wedge,\neg}$ *of* $\mathsf{T}{\models_d}$ *to formulae containing conjunctions and negations. To this end, we note that the cone defined by* $\models_d$ *over the diagram defined by* 1_X*,* d *and the two projections defining the membership relation is (weakly) limiting. Then, the preservation by* T *of weak limits of* W*-shaped diagrams results in the cone defined by* $\mathsf{T}{\models_d}$ *over the diagram defined by* $1_{\mathsf{T}X}$*,* $\mathsf{T}d$ *and the images under* T *of the two previously-mentioned* *projections also being weakly limiting:*

$$
\begin{tikzcd}
& \models_d \arrow[dl, dashed] \arrow[dr, dashed] \arrow[d, dashed] & \\
X \arrow[d, "1_X"'] & \in \arrow[dl] \arrow[dr] & \mathcal{L} \arrow[d, "d"] \\
X & & \mathcal{P}X
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathsf{T}\models_d \arrow[dl, dashed] \arrow[dr, dashed] \arrow[d, dashed] & \\
\mathsf{T}X \arrow[d, "1_{\mathsf{T}X}"'] & \mathsf{T}\in \arrow[dl] \arrow[dr] & \mathsf{T}\mathcal{L} \arrow[d, "\mathsf{T}d"] \\
\mathsf{T}X & & \mathsf{T}\mathcal{P}X
\end{tikzcd}
$$

Hence, t $(\mathsf{T}\models_d)$ ψ is equivalent to t $(\mathsf{T}\in)$ $(\mathsf{T}d)(\psi)$, which in turn is equivalent to $t \in d'(\psi)$, for any $t \in \mathsf{T}X$ and $\psi \in \mathsf{T}\mathcal{L}$. The particular definition of the denotation map d' was driven precisely by the need to ensure that the relations $\models_d \subseteq X \times \mathcal{L}$ and $\models_{d'} \subseteq \mathsf{T}X \times (\mathsf{T}\mathcal{L})^{\wedge,\neg}$ are related as above. Thus, $\mathcal{F}_\mathsf{T}$ captures one step in the definition of the semantics of coalgebraic logic, as defined in [11]. We conclude this example by noting that the preservation of weak pullbacks by T played a crucial rôle in the definition of $\mathcal{F}_\mathsf{T}$.

In the case of inductively-defined coalgebraic types T, T-semantics for the language constructors S_T considered in Example 10 can also be defined inductively.

Example 19 (Inductively-Defined T-Semantics) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ and $\mathsf{S}_\mathsf{T} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ be as in Example 10. A T-semantics $\mathcal{F}_\mathsf{T}$: $\mathsf{IntLang} \to \mathsf{IntLang}$ for S_T is defined inductively as follows:*

- *$\mathcal{F}_A$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_A\mathcal{L}, A, (\{\_\}_A)^\#\rangle$, for any non-empty set A;*
- *$\mathcal{F}_{\mathsf{Id}}$ takes $\langle \mathcal{L}, X, d\rangle$ to itself;*
- *$\mathcal{F}_{\mathcal{P}_\kappa}$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_{\mathcal{P}_\kappa}\mathcal{L}, \mathcal{P}_\kappa X, (d')^\#\rangle$, where $d' : \{\, \Box\varphi \mid \varphi \in \mathcal{L}\,\} \to \mathcal{P}\mathcal{P}_\kappa X$ takes $(\Box\varphi)$ to $\{\, Y \in \mathcal{P}_\kappa X \mid Y \subseteq d(\varphi)\,\}$ for $\varphi \in \mathcal{L}$.*
- *$\mathcal{F}_{\mathcal{D}_\omega}$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_{\mathcal{D}_\omega}\mathcal{L}, \mathcal{D}_\omega X, (d')^\#\rangle$, where $d' : \{\, \Diamond_p\varphi \mid p \in [0,1]\,, \varphi \in \mathcal{L}\,\} \to \mathcal{P}\mathcal{D}_\omega X$ takes $\Diamond_p\varphi$ to $\{\, \mu : X \to [0,1] \mid \sum_{x \models_d \varphi} \mu(x) \geq p \,\}$ for $p \in [0,1]$ and $\varphi \in \mathcal{L}$;*
- *If $\mathcal{F}_{\mathsf{T}_i}$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_{\mathsf{T}_i}\mathcal{L}, \mathsf{T}_i X, d_i\rangle$ for $i \in \{1,2\}$, then:*
  - *$\mathcal{F}_{\mathsf{T}_1+\mathsf{T}_2}$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_{\mathsf{T}_1+\mathsf{T}_2}\mathcal{L}, (\mathsf{T}_1+\mathsf{T}_2)X, ([\mathcal{P}\iota_1, \mathcal{P}\iota_2] \circ (d_1+d_2))^\#\rangle$;*
  - *$\mathcal{F}_{\mathsf{T}_1\times\mathsf{T}_2}$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_{\mathsf{T}_1\times\mathsf{T}_2}\mathcal{L}, (\mathsf{T}_1\times\mathsf{T}_2)X, e \circ (d_1 \times d_2))^\#\rangle$, where the function $e : \mathcal{P}(\mathsf{T}_1X) \times \mathcal{P}(\mathsf{T}_2X) \to \mathcal{P}(\mathsf{T}_1X \times \mathsf{T}_2X)$ takes $\langle Y_1, Y_2\rangle$ to $\{\, \langle y_1, y_2\rangle \mid y_i \in Y_i \text{ for } i = 1,2\,\}$;*
  - *$\mathcal{F}_{(\mathsf{T}_1)^A}$ takes $\langle \mathcal{L}, X, d\rangle$ to $\langle \mathsf{S}_{(\mathsf{T}_1)^A}\mathcal{L}, (\mathsf{T}_1X)^A, (e' \circ \prod_{a\in A} d_1)^\#\rangle$, where the function $e' : \prod_{a\in A} \mathcal{P}(\mathsf{T}_1X) \to \mathcal{P}(\prod_{a\in A}(\mathsf{T}_1X))$ takes $(Y_a)_{a\in A}$ to $\{\, (y_a)_{a\in A} \mid y_a \in Y_a \text{ for } a \in A\,\}$;*
  - *$\mathcal{F}_{\mathsf{T}_1\circ\mathsf{T}_2} ::= \mathcal{F}_{\mathsf{T}_1} \circ \mathcal{F}_{\mathsf{T}_2}$.*

The intuition behind the definitions of $\mathcal{F}_{\mathsf{T}_1+\mathsf{T}_2}$ and $\mathcal{F}_{\mathsf{T}_1\times\mathsf{T}_2}$ are as follows. If $\models_i \subseteq \mathsf{T}_iX \times \mathsf{S}_{\mathsf{T}_i}\mathcal{L}$ is the relation induced by the denotation map $d_i : \mathsf{S}_{\mathsf{T}_i}\mathcal{L} \to$

$\mathcal{P}\mathsf{T}_i X$, for $i = 1, 2$, then the relations $\models_+ \subseteq (\mathsf{T}_1 + \mathsf{T}_2)X \times \mathsf{S}_{\mathsf{T}_1+\mathsf{T}_2}\mathcal{L}$ and $\models_\times \subseteq (\mathsf{T}_1 \times \mathsf{T}_2)X \times \mathsf{S}_{\mathsf{T}_1\times\mathsf{T}_2}\mathcal{L}$ induced by the images of $d : \mathcal{L} \to \mathcal{P}X$ under $\mathcal{F}_{\mathsf{T}_1+\mathsf{T}_2}$ and $\mathcal{F}_{\mathsf{T}_1\times\mathsf{T}_2}$, respectively, are given by:

$$\begin{aligned} \iota_i(t_i) \models_+ \langle\kappa_i\rangle\varphi_i \quad &\textit{iff} \quad t_i \models_i \varphi_i \\ \langle t_1, t_2\rangle \models_\times (\varphi_1, \varphi_2) \quad &\textit{iff} \quad t_1 \models \varphi_1 \textit{ and } t_2 \models \varphi_2 \end{aligned}$$

The definition of $\mathcal{F}_{(\mathsf{T}_1)^A}$ has a similar interpretation to that of $\mathcal{F}_{\mathsf{T}_1\times\mathsf{T}_2}$. Finally, the definition of $\mathcal{F}_{\mathsf{T}_1\circ\mathsf{T}_2}$ can be regarded as introducing a new sort for formulae. The application of $\mathcal{F}_{\mathsf{T}_2}$ yields formulae of this intermediary sort, which are interpreted over $\mathsf{T}_2 X$, whereas the subsequent application of $\mathcal{F}_{\mathsf{T}_1}$ yields formulae which are interpreted over $\mathsf{T}_1\mathsf{T}_2 X$. Further intuitions for this many-sorted structure on formulae will be provided by Examples 42 and 45.

Finally, it is also possible to define a semantics for the language constructor induced by a set of predicate liftings.

Example 20 (T-Semantics for S_Λ) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$, let Λ be a set of predicate liftings for T, and let $\mathsf{S}_\Lambda : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ be as in Example 11. A T-semantics $\mathcal{F}_\Lambda : \mathsf{IntLang} \to \mathsf{IntLang}$ for S_Λ is defined by:*

- *$\mathcal{F}_\Lambda\langle\mathcal{L}, X, d\rangle = \langle\mathsf{S}_\Lambda\mathcal{L}, \mathsf{T}X, (d')^\#\rangle$, where $d' : \{\, [\lambda]\varphi \mid \lambda \in \Lambda,\ \varphi \in \mathcal{L} \,\} \to \mathcal{P}\mathsf{T}X$ takes $[\lambda]\varphi$ to $\lambda_X(d(\varphi))$ for $\lambda \in \Lambda$ and $\varphi \in \mathcal{L}$;*
- *$\mathcal{F}_\Lambda\langle l, f\rangle = \langle\mathsf{S}_\Lambda l, \mathsf{T}f\rangle$.*

5 Strong Expressiveness

This section is concerned with conditions which will later ensure that the language for coalgebras induced by a choice of a language constructor and of an associated semantics is expressive.

The notion of expressiveness introduced below is non-standard in two ways: firstly, it refers to an interpreted language with a fixed semantic domain, and secondly, it is somewhat stronger than the standard notion of expressiveness (which, in this setting, would amount to any two logically-equivalent elements of the semantic domain being equal).

Definition 21 (Strong Expressiveness) *An interpreted language $\langle\mathcal{L}, X, d\rangle$ is **strongly expressive** if for any $x \in X$, there exists $\varphi_x \in \mathcal{L}$ such that $d(\varphi_x) = \{x\}$.*

We immediately note that, if $\langle\mathcal{L}, X, d\rangle$ is strongly expressive, and if $x, y \in X$ are logically equivalent, then $y \models_d \varphi_x$, and hence $y = x$.

Remark 22 *Equivalently, $\langle \mathcal{L}, X, d\rangle$ is strongly expressive if and only if there exists a function $i : X \to \mathcal{L}$ such that $d \circ i = \{\_\}_X$:*

$$\begin{tikzcd} & \mathcal{L} \arrow[d, "d"] \\ X \arrow[ur, dashed, "i"] \arrow[r, "\{\_\}_X"'] & \mathcal{P}X \end{tikzcd}$$

with the natural transformation $\{\_\} : \mathsf{Id} \Rightarrow \mathcal{P}$ being given by $\{\_\}_S(s) = \{s\}$ for $s \in S$ and $S \in |\mathsf{Set}|$. This equivalent formulation will prove more useful in what follows. We also note that, since $\{\_\}_X$ is injective, any function i satisfying the above condition is itself injective.

The next definition captures a condition which will allow us to prove that the (yet to be defined) language for coalgebras induced by a choice of a language constructor and of an associated semantics is expressive (in the sense of Definition 5).

Definition 23 *A T-semantics $\mathcal{F}$ for a language constructor S* ***preserves expressiveness*** *if $\mathcal{F}\langle \mathcal{L}, X, d\rangle$ is strongly expressive whenever $\langle \mathcal{L}, X, d\rangle$ is.*

That is, $\mathcal{F}$ preserves expressiveness if whenever one starts with a language $\mathcal{L}$ which is characterising for a set X, by applying $\mathcal{F}$ one obtains a language $\mathsf{S}\mathcal{L}$ which is characterising for the set $\mathsf{T}X$.

All the T-semantics defined in Section 4 preserve expressiveness, as shown in the following.

Example 24 *If $\mathcal{F}_\mathsf{T} : \mathsf{IntLang} \to \mathsf{IntLang}$ is as in Example 18, then $\mathcal{F}_\mathsf{T}$ preserves expressiveness. For, if the left triangle below commutes, so does the top-right triangle.*

$$\begin{tikzcd} & \mathcal{L} \arrow[d, "d"] \\ X \arrow[ur, dashed, "i"] \arrow[r, "\{\_\}_X"'] & \mathcal{P}X \end{tikzcd} \qquad \begin{tikzcd} & \mathsf{T}\mathcal{L} \arrow[d, "\mathsf{T}d"] \\ \mathsf{T}X \arrow[ur, dashed, "\mathsf{T}i"] \arrow[r, "\mathsf{T}\{\_\}_X"'] \arrow[dr, "\{\_\}_{\mathsf{T}X}"'] & \mathsf{T}\mathcal{P}X \arrow[d, "\epsilon_X"] \\ & \mathcal{P}\mathsf{T}X \end{tikzcd}$$

Also, the bottom-right triangle commutes. For, $t' \in \epsilon_X((\mathsf{T}\{\_\}_X)(t))$ translates to t' $(\mathsf{T}\in)$ $(\mathsf{T}\{\_\}_X)(t)$. But the fact that the left diagram below is weakly limiting together with the preservation by T of weak limits of W-shaped diagrams (see

Remark 1) result in the right diagram also being weakly limiting:

$$
\begin{tikzcd}
 & X \arrow[dl, "1_X"'] \arrow[dr, "1_X"] \arrow[dd, dashed] & \\
X \arrow[dd, "1_X"'] & & X \arrow[dd, "\{\_\}_X"] \\
 & \in \arrow[dl] \arrow[dr] & \\
X & & \mathcal{P}X
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathsf{T}X \arrow[dl, "1_{\mathsf{T}X}"'] \arrow[dr, "1_{\mathsf{T}X}"] \arrow[dd, dashed] & \\
\mathsf{T}X \arrow[dd, "1_{\mathsf{T}X}"'] & & \mathsf{T}X \arrow[dd, "\mathsf{T}\{\_\}_X"] \\
 & \mathsf{T}\!\in \arrow[dl] \arrow[dr] & \\
\mathsf{T}X & & \mathsf{T}\mathcal{P}X
\end{tikzcd}
$$

Thus, t' $(\mathsf{T}\!\in)$ $(\mathsf{T}\{\_\}_X)(t)$ is equivalent to $t' = t$. Hence, $\epsilon_X((\mathsf{T}\{\_\}_X)(t)) = \{t\}$. It then follows that $\epsilon_X \circ \mathsf{T}d \circ \mathsf{T}i = \{\_\}_{\mathsf{T}X}$, that is, the function $\mathsf{T}i : \mathsf{T}X \to \mathsf{T}\mathcal{L} \subseteq \mathsf{S}_\mathsf{T}\mathcal{L}$ satisfies the condition in Remark 22.

Proposition 25 *$\mathcal{F}_A$, $\mathcal{F}_{\mathsf{Id}}$, $\mathcal{F}_{\mathcal{P}_\kappa}$ and $\mathcal{F}_{\mathcal{D}_\omega}$, as defined in Example 19, preserve expressiveness.*

PROOF (Sketch). The fact that $\mathcal{F}_A$ and $\mathcal{F}_{\mathsf{Id}}$ preserve expressiveness follows immediately from their definition. To see that $\mathcal{F}_{\mathcal{P}_\kappa}$ and $\mathcal{F}_{\mathcal{D}_\omega}$ preserve expressiveness, let $\langle \mathcal{L}, X, d\rangle$ be strongly expressive, let $i : X \to \mathcal{L}$ satisfy the condition in Remark 22, and define:

- $i_{\mathcal{P}_\kappa} : \mathcal{P}_\kappa X \to \mathsf{S}_{\mathcal{P}_\kappa}\mathcal{L},\ \ Y \ \mapsto\ \Box(\bigvee_{y\in Y} i(y)) \wedge \bigwedge_{y \in Y} \Diamond i(y)$;
- $i_{\mathcal{D}_\omega} : \mathcal{D}_\omega X \to \mathsf{S}_{\mathcal{D}_\omega}\mathcal{L},\ \ \mu \ \mapsto\ \bigwedge_{\mu(x)\neq 0} \Diamond_{\mu(x)} i(x)$;

We note that both the conjunction and the disjunction used in the definition of $i_{\mathcal{P}_\kappa}$ have size smaller than κ. An easy calculation shows that $i_{\mathcal{P}_\kappa}$ makes $\mathcal{F}_{\mathcal{P}_\kappa}\langle \mathcal{L}, X, d\rangle$ strongly expressive. As for $\mathcal{F}_{\mathcal{D}_\omega}\langle \mathcal{L}, X, d\rangle = \langle \mathsf{S}_{\mathcal{D}_\omega}\mathcal{L}, \mathcal{D}_\omega X, d'\rangle$, the definitions of $d' : \mathsf{S}_{\mathcal{D}_\omega}\mathcal{L} \to \mathcal{P}\mathcal{D}_\omega X$ (see Example 19) and $i_{\mathcal{D}_\omega}(\mu) \in \mathsf{S}_{\mathcal{D}_\omega}\mathcal{L}$ immediately yield $\mu \in d'(i_{\mathcal{D}_\omega}(\mu))$. Also, whenever $\mu' \in d'(i_{\mathcal{D}_\omega}(\mu))$, we have $\sum_{x' \in d(i(x))} \mu'(x') \geq \mu(x)$, or equivalently $\mu'(x) \geq \mu(x)$, for all $x \in X$. This, together with $\sum_{x\in X} \mu'(x) = 1 = \sum_{x \in X} \mu(x)$ gives $\mu'(x) = \mu(x)$ for all $x \in X$, and hence $\mu' = \mu$. Thus, $\mathcal{F}_{\mathcal{D}_\omega}\langle \mathcal{L}, X, d\rangle$ is strongly expressive.

Proposition 26 *Let $\mathsf{T}_1, \mathsf{T}_2 : \mathsf{Set} \to \mathsf{Set}$, let $\mathsf{S}_{\mathsf{T}_1}, \mathsf{S}_{\mathsf{T}_2} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ be language constructors, and let $\mathcal{F}_{\mathsf{T}_i}$ be a T_i-semantics for $\mathsf{S}_{\mathsf{T}_i}$, for $i = 1, 2$. If $\mathcal{F}_{\mathsf{T}_1}$ and $\mathcal{F}_{\mathsf{T}_2}$ preserve expressiveness, then so do $\mathcal{F}_{\mathsf{T}_1+\mathsf{T}_2}$, $\mathcal{F}_{\mathsf{T}_1\times\mathsf{T}_2}$, $\mathcal{F}_{(\mathsf{T}_1)^A}$ and $\mathcal{F}_{\mathsf{T}_1\circ\mathsf{T}_2}$, as defined in Example 19.*

PROOF (Sketch). Let $\langle \mathcal{L}, X, d\rangle$ be a strongly expressive interpreted language, and let $\mathcal{F}_{\mathsf{T}_i}\langle \mathcal{L}, X, d\rangle = \langle \mathsf{S}_{\mathsf{T}_i}\mathcal{L}, \mathsf{T}_i X, d_i\rangle$, for $i \in \{1, 2\}$. Also, let $i_1 : \mathsf{T}_1 X \to \mathsf{S}_{\mathsf{T}_1}\mathcal{L}$ and $i_2 : \mathsf{T}_2 X \to \mathsf{S}_{\mathsf{T}_2}\mathcal{L}$ satisfy the condition in Remark 22, and define:

- $i_+ : \mathsf{T}_1 X + \mathsf{T}_2 X \to \mathsf{S}_{\mathsf{T}_1+\mathsf{T}_2}\mathcal{L},\ \ \iota_j(t_j) \ \mapsto\ \langle \kappa_j \rangle i_j(t_j)$ for $j = 1, 2$;
- $i_\times : \mathsf{T}_1 X \times \mathsf{T}_2 X \to \mathsf{S}_{\mathsf{T}_1 \times \mathsf{T}_2}\mathcal{L},\ \ \langle t_1, t_2 \rangle \ \mapsto\ \langle i_1(t_1), i_2(t_2) \rangle$
- $i_{(\_)^A} : (\mathsf{T}_1 X)^A \to \mathsf{S}_{(\mathsf{T}_1)^A}\mathcal{L},\ \ t \mapsto (i_1(t(a))_{a \in A}$

Some straightforward calculations show that these functions also satisfy the condition in Remark 22. Finally, the fact that $\mathcal{F}_{\mathsf{T}_1 \circ \mathsf{T}_2}$ preserves expressiveness follows immediately from its definition, and from the fact that each of $\mathcal{F}_{\mathsf{T}_1}$ and $\mathcal{F}_{\mathsf{T}_2}$ preserves expressiveness.

Propositions 25 and 26 now yield, for each endofunctor T of the form considered in Example 19, a T-semantics $\mathcal{F}_\mathsf{T}$ for S_T which preserves expressiveness.

Corollary 27 *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *and* $\mathcal{F}_\mathsf{T} : \mathsf{IntLang} \to \mathsf{IntLang}$ *be as in Example 19. Then,* $\mathcal{F}_\mathsf{T}$ *preserves expressiveness.*

Finally, in the case of a separating set of predicate liftings Λ, the associated T-semantics $\mathcal{F}_\Lambda$ preserves expressiveness.

Example 28 *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *be an accessible endofunctor, let* Λ *be a separating set of predicate liftings for* T*, and let* S_Λ *and* $\mathcal{F}_\Lambda$ *be as in Example 20. Then, the* T*-semantics* $\mathcal{F}_\Lambda$ *for* S_Λ *preserves expressiveness, for a suitable choice of* κ*. To see this, let* $\langle \mathcal{L}, X, d \rangle$ *be strongly expressive, let* $i : X \to \mathcal{L}$ *satisfy the condition in Remark 22, and let* $\mathcal{F}_\Lambda \langle \mathcal{L}, X, d \rangle = \langle \mathsf{S}_\Lambda \mathcal{L}, \mathsf{T}X, d' \rangle$*. Now define* $i' : \mathsf{T}X \to \mathsf{S}_\Lambda \mathcal{L}$ *by:*

$$i'(t) = \bigwedge_{\substack{\lambda \in \Lambda \\ Y \in \mathcal{P}X \\ \lambda_X(Y) \ni t}} [\lambda]\varphi_Y \ \wedge \bigwedge_{\substack{\lambda \in \Lambda \\ Y \in \mathcal{P}X \\ \overline{\lambda_X(\overline{Y}) \ni t}}} \langle \lambda \rangle \varphi_Y\,, \quad t \in \mathsf{T}X$$

with φ_Y *being given by* $\bigvee_{y \in Y} i(y)$ *for any* $Y \in \mathcal{P}X$ [7]*. It is shown in [12, Section 7] that both the disjunctions defining the* φ_Y*s and the two conjunctions defining* $i'(t)$ *can be brought down to a size smaller than some fixed cardinal* κ*, with* κ *depending only on* $\mathsf{rank}(\mathsf{T})$ *and* $\mathsf{card}(\Lambda)$*. It is precisely this* κ *that we are considering in the definition of* $\mathcal{F}_\Lambda$*. The definition of* $d' : \mathsf{S}_\Lambda \mathcal{L} \to \mathcal{P}\mathsf{T}X$ *and the fact that* $d(\varphi_Y) = \bigcup_{y \in Y} d(i(y)) = \bigcup_{y \in Y} \{y\} = Y$ *for any* $Y \in \mathcal{P}X$ *give* $t \in d'(i'(t))$*. Now assume* $t' \neq t$*. Then, by* Λ *being saturated, one of the following is true:*

(1) there exist $\lambda \in \Lambda$ *and* $Y \in \mathcal{P}X$ *such that* $t \in \lambda_X(Y)$ *but* $t' \notin \lambda_X(Y)$*;*

(2) there exist $\lambda \in \Lambda$ *and* $Y \in \mathcal{P}X$ *such that* $t' \in \lambda_X(Y)$ *but* $t \notin \lambda_X(Y)$*.*

---

[7] Note that the closure of $\mathcal{L}$ under $\bigwedge$ and $\neg$ (and hence also under $\bigvee$) gives $\varphi_Y \in \mathcal{L}$, and therefore $i'(t) \in \mathcal{L}'$.

Depending on which of these holds, either $[\lambda]\varphi_Y$ or $\langle\lambda\rangle\varphi_{\overline{Y}}$ does not hold in t', while $t \in \lambda_X(Y)$ and respectively $t \in \overline{\lambda_X(Y)}$ holds. Hence, $t' \notin d'(i'(t))$. This concludes the proof of the fact that $\mathcal{F}_\Lambda$ preserves expressiveness.

6 Expressive Languages for Coalgebras, Compositionally

We now use the notions defined in previous sections to derive languages for coalgebras in an inductive fashion. We show that each pair consisting of a language constructor and associated T-semantics induces a language for T-coalgebras, and, provided that the T-semantics preserves expressiveness, the resulting language for coalgebras is expressive (in the sense of Definition 5). To this end, we consider the following notion of language for T-coalgebras, which, as we will show, specialises Definition 5 by making implicit the adequacy of the language w.r.t. behavioural equivalence.

Definition 29 (Language for T-Coalgebras) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$, and let $\mathsf{U} : \mathsf{Coalg}(\mathsf{T}) \to \mathsf{Set}$ denote the functor taking T-coalgebras to their carrier. A* ***language for T-coalgebras*** *is a pair $\langle \mathcal{L}, \mathsf{d}\rangle$, with $\mathcal{L}$ a B_κ-algebra and $\mathsf{d} : \mathcal{L} \Rightarrow \check{\mathcal{P}} \circ \mathsf{U}$ a natural transformation*[8]*. Also, a* ***map between languages*** *$\langle \mathcal{L}, \mathsf{d}\rangle$ and $\langle \mathcal{L}', \mathsf{d}'\rangle$ is a B_κ-algebra homomorphism $l : \mathcal{L} \to \mathcal{L}'$, such that $\mathsf{d}' \circ l = \mathsf{d}$*[9]*.*

The above definition imposes the structure of a B_κ-algebra to the set of formulae of a language for T-coalgebras. Given such a language $\langle \mathcal{L}, \mathsf{d}\rangle$, the components $d_\gamma : \mathcal{L} \to \mathcal{P}C$ of the natural transformation d provide interpretations of the formulae in $\mathcal{L}$ over the carriers C of all T-coalgebras $\langle C, \gamma\rangle$. By writing $c \models_\gamma \varphi$ for $c \in \mathsf{d}_\gamma(\varphi)$, with $c \in C$ and $\varphi \in \mathcal{L}$, we obtain a language for T-coalgebras in the sense of Definition 5. Moreover, the naturality of d ensures that this language is adequate. For, given T-coalgebras $\langle C, \gamma\rangle$ and $\langle D, \delta\rangle$, if $c \in C$ and $d \in D$ are behaviourally equivalent, and hence identified by some T-coalgebra homomorphisms $f : \langle C, \gamma\rangle \to \langle E, \eta\rangle$ and $g : \langle D, \delta\rangle \to \langle E, \eta\rangle$, then $c \in \mathsf{d}_\gamma(\varphi)$ if and only if $f(c) = g(d) \in \mathsf{d}_\eta(\varphi)$ if and only if $d \in \mathsf{d}_\delta(\varphi)$, for any $\varphi \in \mathcal{L}$. We also note that, in the presence of a final T-coalgebra $\langle Z, \zeta\rangle$, $\langle \mathcal{L}, \mathsf{d}\rangle$ is fully determined by d_ζ: if $!_\gamma : \langle C, \gamma\rangle \to \langle Z, \zeta\rangle$ is the unique T-coalgebra homomorphism from the T-coalgebra $\langle C, \gamma\rangle$ to the final one, then $\mathsf{d}_\gamma = \hat{\mathcal{P}}\mathsf{U}!_\gamma \circ \mathsf{d}_\zeta$.

Remark 30 *For each regular cardinal α, we can derive a language $\langle \mathcal{L}, \mathsf{d}\rangle$ for T-coalgebras from an interpreted language of form $\langle \mathcal{L}, Z_\alpha, d\rangle$, with Z_α being as*

[8] Here $\mathcal{L}$ denotes the constant functor $\langle C, \gamma\rangle \mapsto \mathcal{L}$.

[9] $l : \mathcal{L} \Rightarrow \mathcal{L}'$ denotes the constant natural transformation all of whose components are given by the B_κ-algebra homomorphism l.

in Definition 2. Specifically, for a T*-coalgebra* $\langle C, \gamma \rangle$*, we let* $\mathsf{d}_\gamma = \hat{\mathcal{P}}\gamma_\alpha \circ d$*:*

$$
\begin{tikzcd}
\mathcal{L} \arrow[d, "d"'] & \\
\mathcal{P}Z_\alpha \arrow[r, "\hat{\mathcal{P}}\gamma_\alpha"] & \mathcal{P}C
\end{tikzcd}
$$

with $\gamma_\alpha : C \to Z_\alpha$ *being as in Remark 3. The fact that* T*-coalgebra homomorphisms* $f : \langle C, \gamma \rangle \to \langle D, \delta \rangle$ *define morphisms of cones* $f : (\gamma_\alpha) \to (\delta_\alpha)$ *ensures the naturality of* d*.*

The languages which interest us are those obtained by taking $\alpha = \kappa$ in Remark 30, where $\kappa \geq \mathsf{rank}(\mathsf{T})$. For, in this case, if the interpreted language $\langle \mathcal{L}, Z_\kappa, d \rangle$ is strongly expressive, then the induced language $\langle \mathcal{L}, \mathsf{d} \rangle$ is expressive.

Proposition 31 *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *be a* κ*-accessible endofunctor, and let* $\langle \mathcal{L}, Z_\kappa, d \rangle$ *be strongly expressive. Then, the induced language* $\langle \mathcal{L}, \mathsf{d} \rangle$ *for* T*-coalgebras is expressive.*

PROOF. Let $\langle C, \gamma \rangle$ and $\langle E, \eta \rangle$ be T-coalgebras, and let $c \in C$ and $e \in E$ be such that $c \models_\gamma \varphi$ if and only if $e \models_\eta \varphi$, for any $\varphi \in \mathcal{L}$. By the definition of $\langle \mathcal{L}, \mathsf{d} \rangle$, $c \models_\gamma \varphi$ precisely when $\gamma_\kappa(c) \in d(\varphi)$. Hence, $c \models_\gamma i(\gamma_\kappa(c))$, with $i : Z_\kappa \to \mathcal{L}$ being as in Remark 22. But then $e \models_\eta i(\gamma_\kappa(c))$, or equivalently, $\eta_\kappa(e) \in d(i(\gamma_\kappa(c))) = \{\gamma_\kappa(c)\}$. Thus, $\eta_\kappa(e) = \gamma_\kappa(c)$. It then follows from the definition of Z_κ together with Remark 4 that c and e are behaviourally equivalent.

Next, we show how to derive a strongly expressive interpreted language for Z_κ, and hence an expressive language for T-coalgebras, in the case when T is κ-accessible. Specifically, we define an ordinal-indexed sequence of interpreted languages, one for each element of the final sequence of T.

Definition 32 (Language Sequence) *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$*, let* $\mathsf{S} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *be a language constructor, and let* $\mathcal{F} : \mathsf{IntLang} \to \mathsf{IntLang}$ *be a* T*-semantics for* S*. The* ***language sequence induced by*** $\mathcal{F}$ *is the initial sequence*[10] *of* $\mathcal{F}$*.*

That is, the language sequence induced by $\mathcal{F}$ is an ordinal-indexed sequence of interpreted languages $(\langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle)$, together with a family $(\langle \iota^\alpha_\beta, p^\alpha_\beta \rangle)_{\beta \leq \alpha}$ of maps $\langle \iota^\alpha_\beta, p^\alpha_\beta \rangle : \langle \mathcal{L}_\beta, Z_\beta, d_\beta \rangle \to \langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle$, satisfying:

- $\langle \mathcal{L}_{\alpha+1}, Z_{\alpha+1}, d_{\alpha+1} \rangle = \mathcal{F} \langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle$

[10] The initial sequence of an endofunctor is defined similarly to its final sequence.

- $\langle \iota^{\alpha+1}_{\beta+1}, p^{\alpha+1}_{\beta+1} \rangle = \mathcal{F}\langle \iota^{\alpha}_{\beta}, p^{\alpha}_{\beta} \rangle$ for $\beta \leq \alpha$
- $\langle \iota^{\alpha}_{\alpha}, p^{\alpha}_{\alpha} \rangle = 1_{\langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle}$
- $\langle \iota^{\alpha}_{\gamma}, p^{\alpha}_{\gamma} \rangle = \langle \iota^{\alpha}_{\beta}, p^{\alpha}_{\beta} \rangle \circ \langle \iota^{\beta}_{\gamma}, p^{\beta}_{\gamma} \rangle$ for $\gamma \leq \beta \leq \alpha$
- if α is a limit ordinal, the cocone $\langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle$, $(\langle \iota^{\alpha}_{\beta}, p^{\alpha}_{\beta} \rangle)_{\beta<\alpha}$ for $(\langle \iota^{\beta}_{\gamma}, p^{\beta}_{\gamma} \rangle)_{\gamma \leq \beta < \alpha}$ is colimiting [11] .

An immediate observation is that the $\mathsf{Alg}(\mathsf{B}_\kappa)$- and Set-sequences underlying the language sequence induced by $\mathcal{F}$ are the initial sequence of S and the final sequence of T, respectively.

Proposition 33 *Let $(\langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle), (\langle \iota^{\alpha}_{\beta}, p^{\alpha}_{\beta} \rangle)_{\beta \leq \alpha}$ be the language sequence induced by a T-semantics $\mathcal{F}$ for S. Then, $(\mathcal{L}_\alpha), (\iota^{\alpha}_{\beta})_{\beta \leq \alpha}$ is the initial sequence of S, while $(Z_\alpha), (p^{\alpha}_{\beta})_{\beta \leq \alpha}$ is the final sequence of T.*

PROOF. The fact that $\mathcal{F}$ is a T-semantics for S yields $\mathcal{L}_{\alpha+1} = \mathsf{S}\mathcal{L}_\alpha$ and $Z_{\alpha+1} = \mathsf{T}Z_\alpha$ for any α, as well as $\iota^{\alpha+1}_{\beta+1} = \mathsf{S}\iota^{\alpha}_{\beta}$ and $p^{\alpha+1}_{\beta+1} = \mathsf{T}p^{\alpha}_{\beta}$ for any $\beta \leq \alpha$. If α is a limit ordinal, it follows by Proposition 16 that the colimit of the diagram defined by $(\langle \iota^{\beta}_{\gamma}, p^{\beta}_{\gamma} \rangle)_{\gamma \leq \beta < \alpha}$ is constructed by first computing the limit in Set of $(p^{\beta}_{\gamma})_{\gamma \leq \beta < \alpha}$ (with limit object Z_α), then using the coreindexing functors $(p^{\alpha}_{\beta})_* : \mathsf{IntLang}_{Z_\beta} \to \mathsf{IntLang}_{Z_\alpha}$ to obtain a diagram in $\mathsf{IntLang}_{Z_\alpha}$, and finally computing the limit of this diagram in $\mathsf{IntLang}_{Z_\alpha}$. It thus follows immediately that the cone $Z_\alpha, (p^{\alpha}_{\beta})_{\beta<\alpha}$ for $(p^{\beta}_{\gamma})_{\gamma \leq \beta < \alpha}$ is limiting. Also, the definition of the coreindexing functors together with the construction of colimits in $\mathsf{IntLang}_{Z_\alpha}$ from colimits in $\mathsf{Alg}(\mathsf{B}_\kappa)$ result in the cocone $\mathcal{L}_\alpha, (\iota^{\alpha}_{\beta})_{\beta<\alpha}$ for $(\iota^{\beta}_{\gamma})_{\gamma \leq \beta < \alpha}$ being colimiting. This concludes the proof.

Remark 34 *The existence of maps $\langle \iota^{\alpha}_{\beta}, p^{\alpha}_{\beta} \rangle : \langle \mathcal{L}_\beta, Z_\beta, d_\beta \rangle \to \langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle$ with $\beta \leq \alpha$ amounts to the commutativity of diagrams of form:*

$$\begin{tikzcd}
\mathcal{L}_\beta \arrow[r, "\iota^{\alpha}_{\beta}"] \arrow[d, "d_\beta"'] & \mathcal{L}_\alpha \arrow[d, "d_\alpha"] \\
\mathcal{P}Z_\beta \arrow[r, "\hat{\mathcal{P}}p^{\alpha}_{\beta}"'] & \mathcal{P}Z_\alpha
\end{tikzcd}$$

Our next result concerns the strong expressiveness of the languages belonging to the language sequence induced by a T-semantics which preserves expressiveness.

Theorem 35 (Strong Expressiveness for Language Sequence) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$, let $\mathsf{S} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ be a language constructor, and let*

[11] Recall from Proposition 16 that $\mathsf{IntLang}$ has all colimits.

$\mathcal{F}$: $\mathsf{IntLang} \to \mathsf{IntLang}$ *be a* T*-semantics for* S*. If* $\mathcal{F}$ *preserves expressiveness, then* $\langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle$ *is strongly expressive for any* $\alpha \leq \kappa$ [12] *.*

PROOF. We use transfinite induction to prove this statement.

If $\alpha = \beta + 1$, the fact that $\langle \mathcal{L}_\beta, Z_\beta, d_\beta \rangle$ is strongly expressive together with the fact that $\mathcal{F}$ preserves expressiveness result in $\langle \mathcal{L}_\alpha, Z_\alpha, d_\alpha \rangle = \mathcal{F}\langle \mathcal{L}_\beta, Z_\beta, d_\beta \rangle$ being strongly expressive.

If α is a limit ordinal, we define $i_\alpha : Z_\alpha \to \mathcal{L}_\alpha$ by:

$$i_\alpha(x) = \bigwedge_{\beta < \alpha} \iota^\alpha_\beta(i_\beta(p^\alpha_\beta(x))) \quad \text{for } x \in Z_\alpha$$

once each $i_\beta : Z_\beta \to \mathcal{L}_\beta$ with $\beta < \alpha$ has been defined. In particular, $i_0 \equiv \top$. We note that, since $\alpha \leq \kappa$, the size of the conjunction used in the definition of i_α is smaller than κ. Then, the fact that $d_\alpha \circ i_\alpha = \{\_\}_{Z_\alpha}$ follows from $d_\beta \circ i_\beta = \{\_\}_{Z_\beta}$ for each $\beta < \alpha$, together with Remark 34, the preservation by d_α of the B_κ-structure, and the universal property of $Z_\alpha, (p^\alpha_\beta)_{\beta<\alpha}$. This concludes the proof.

Now recall from Proposition 31 that, if T is κ-accessible, the language $\langle \mathcal{L}, \mathsf{d} \rangle$ for T-coalgebras induced by a strongly expressive interpreted language $\langle \mathcal{L}, Z_\kappa, d \rangle$ is expressive. This prompts the following definition.

Definition 36 (Induced Language for T-Coalgebras) *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ *be a* κ*-accessible endofunctor, let* $\mathsf{S} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *be a language constructor, and let* $\mathcal{F} : \mathsf{IntLang} \to \mathsf{IntLang}$ *be a* T*-semantics for* S*. The* ***language induced by*** $\mathcal{F}$ *is* $\langle \mathcal{L}_\kappa, \mathsf{d}_\kappa \rangle$ *(as defined in Remark 30).*

In the above definition, the cardinal κ is determined by the choice of $\mathcal{F}$, whereas the requirement that T is κ-accessible amounts to $\kappa \geq \mathsf{rank}(\mathsf{T})$. By instantiating T, κ and $\mathcal{F}$, we obtain languages for coalgebras of specific endofunctors T. The languages thus obtained only contain conjunctions and disjunctions of size smaller than κ. In particular, if $\kappa = \omega$, the resulting languages only contain finite conjunctions and disjunctions.

Theorem 35 together with Proposition 31 now yield the following (Hennessy-Milner style) characterisation result for behavioural equivalence.

Corollary 37 (Expressiveness of Induced Language) *Let* $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$*,* $\mathsf{S} : \mathsf{Alg}(\mathsf{B}_\kappa) \to \mathsf{Alg}(\mathsf{B}_\kappa)$ *and* $\mathcal{F} : \mathsf{IntLang} \to \mathsf{IntLang}$ *be as in Definition 36. If* $\mathcal{F}$

[12] Recall from Remark 13 that κ is an implicit parameter of the category $\mathsf{IntLang}$, imposing a bound on the size of the conjunctions allowed in B_κ-algebras.

preserves expressiveness, then the language induced by $\mathcal{F}$ is expressive.

The remainder of this section describes some expressive languages for coalgebras, obtained by instantiating T, κ and $\mathcal{F}$ in Corollary 37.

Example 38 (Coalgebraic Logic) *Let T be as in Definition 6, let $\kappa = \mathsf{rank}(\mathsf{T})$, and let $\mathcal{F}_\mathsf{T}$ be as in Example 18. The language induced by $\mathcal{F}_\mathsf{T}$ is a fragment*[13] *of the language of coalgebraic logic [11], enriched with negation. By Corollary 37, this language fragment is expressive. Moreover, a closer inspection of the definition of the functions $i_\alpha : Z_\alpha \to \mathcal{L}_\alpha$ for $\alpha \leq \kappa$ (see Example 24 and the proof of Theorem 35) reveals that negation is not needed to characterise the states of the final T-coalgebra. Its presence in the previously-mentioned language fragment is simply due to the general setting of this paper.*

Example 39 (Languages from Predicate Liftings) *Let T, Λ, κ and $\mathcal{F}_\Lambda$ be as in Example 28. In particular, the cardinal κ is determined by $\mathsf{rank}(\mathsf{T})$ and $\mathsf{card}(\Lambda)$. The language induced by $\mathcal{F}_\Lambda$ coincides with the language defined in [12]. By applying Corollary 37 to $\mathcal{F}_\Lambda$, we obtain an alternative proof of the expressiveness of this language.*

Example 40 (Languages for Finite Kripke Polynomial Functors) *Let $\kappa = \omega$, let T be a finite Kripke polynomial (and hence ω-accessible) endofunctor, and let $\mathcal{F}_\mathsf{T}$ be is as in Example 19. The language induced by $\mathcal{F}_\mathsf{T}$ is essentially the same as the language defined in [9]. The only difference is that here we take formulae of form $\langle \varphi_1, \varphi_2 \rangle$ and $(\varphi_a)_{a \in A}$ as primitive, whereas formulae of form $[\pi_i]\varphi_i$ and $[a]\varphi$ were considered in [9]. By applying Corollary 37 to $\mathcal{F}_\mathsf{T}$, we obtain an alternative proof of [9, Corollary 5.9], which stated the expressiveness of the above-mentioned language.*

Corollary 37 also yields expressive *infinitary* languages for coalgebras of κ-bounded Kripke polynomial functors. The resulting languages contain conjunctions and disjunctions of size smaller than κ. Expressive languages for *κ-bounded* Kripke polynomial functors were not considered in [13,9].

Corollary 41 (Languages for Kripke Polynomial Functors) *Let T be a κ-bounded Kripke polynomial functor, and let $\mathcal{F}_\mathsf{T}$ be is as in Example 19. Then, the language induced by $\mathcal{F}_\mathsf{T}$ is expressive.*

PROOF. The statement follows immediately from Propositions 25 and 26, and Corollary 37.

[13] Note that we only allow conjunctions of size smaller than κ.

In particular, Corollary 41 yields a language for coalgebras of the functor $\mathcal{P}_\kappa \circ \mathcal{P}_\kappa$. This language has a two-sorted structure on formulae, as illustrated by the following example.

Example 42 (Language for $\mathcal{P}_\kappa \circ \mathcal{P}_\kappa$-Coalgebras) *Let $\mathsf{T} = \mathcal{P}_\kappa \circ \mathcal{P}_\kappa$, and let $\mathcal{F}_\mathsf{T}$ be as in Example 19. The language $\mathcal{L}$ induced by $\mathcal{F}_\mathsf{T}$ can be described by the following grammar:*

$$\begin{aligned} \mathcal{L} \ni \varphi &::= \bigwedge \Phi \mid \neg\varphi \mid \Box\psi \qquad (\psi \in \mathcal{L}') \\ \mathcal{L}' \ni \psi &::= \bigwedge \Psi \mid \neg\psi \mid \Box\varphi \qquad (\varphi \in \mathcal{L}) \end{aligned}$$

It is worth noting here that the interleaving of modal operators and propositional connectives is precisely what makes the resulting language expressive. The absence of such an interleaving mechanism in the logics induced by predicate liftings resulted in the approach in [12] failing to provide an expressive language for $\mathcal{P}_\kappa \circ \mathcal{P}_\kappa$-coalgebras.

Finally, Corollary 37 together with the compositionality results in Propositions 25 and 26 can also be used to obtain *finitary* expressive languages for a large class of probabilistic system types.

Corollary 43 (Expressive Languages for Probabilistic Systems) *Let $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$ be an endofunctor generated using the following syntax:*

$$\mathsf{T} ::= A \mid \mathsf{Id} \mid \mathcal{P}_\omega \mid \mathcal{D}_\omega \mid \mathsf{T}_1 + \mathsf{T}_2 \mid \mathsf{T}_1 \times \mathsf{T}_2 \mid (\mathsf{T}_1)^A \mid \mathsf{T}_1 \circ \mathsf{T}_2$$

where all the sets used as exponents are finite. Also, let $\kappa = \omega$, and let $\mathcal{F}_\mathsf{T}$ be as in Example 19. Then, the language induced by $\mathcal{F}_\mathsf{T}$ is both finitary and expressive.

PROOF. The statement follows from the fact that any such endofunctor T is ω-accessible, and from the observation that (the proofs of) Proposition 25 and 26 allow us to take $\kappa = \omega$ in Corollary 37.

Two instances of probabilistic system types covered by this result are described below.

Example 44 (Language for Probabilistic Transition Systems) *Let $\mathsf{T} = (1 + \mathcal{D}_\omega)^A$, let $\kappa = \omega$, and let $\mathcal{F}_\mathsf{T}$ be as in Example 19. The language induced by $\mathcal{F}_\mathsf{T}$ is essentially the language considered in [10]. The closure under conjunction and negation at each step in the modular construction of the language results in the induced language also containing some additional formulae. However, these formulae do not add any expressive power to the language, and can be systematically discarded.*

Example 45 (Languages for Probabilistic Automata) *The probabilistic automata considered in [15] can be modelled as coalgebras of the endofunctors* $\mathsf{T}_1 = \mathcal{P}_\omega \circ (A \times \mathcal{D}_\omega)$ *(simple probabilistic automata) and* $\mathsf{T}_2 = \mathcal{P}_\omega \circ \mathcal{D}_\omega \circ (A \times \mathsf{Id})$ *(general probabilistic automata)*[14]*. By Corollary 43, the languages induced by* $\mathcal{F}_{\mathsf{T}_1}$ *and* $\mathcal{F}_{\mathsf{T}_2}$ *are both finitary and expressive. In the case of simple probabilistic automata, the induced language* $\mathcal{L}$ *can be described using the following grammar*[15]*:*

$$
\begin{array}{rcll}
\mathcal{L} & \ni & \varphi ::= \top \mid \varphi \wedge \varphi' \mid \neg\varphi \mid \Box\psi & (\psi \in \mathcal{L}') \\
\mathcal{L}' & \ni & \psi ::= \top \mid \psi \wedge \psi' \mid \neg\psi \mid (a, \xi) & (a \in A\,,\ \xi \in \mathcal{L}'') \\
\mathcal{L}'' & \ni & \xi ::= \top \mid \xi \wedge \xi' \mid \neg\xi \mid \Diamond_p\varphi & (\varphi \in \mathcal{L})
\end{array}
$$

Again, we note the many-sorted structure on formulae, which is due to the modular construction of the language.

The coalgebraic semantics of $\mathcal{L}$ *is given by:*

$$c \models_\gamma \varphi \quad \textit{iff} \quad \gamma(c) \models \varphi$$

where the relations

$$
\begin{array}{rcl}
\models & \subseteq & \mathcal{P}_\omega(A \times \mathcal{D}_\omega C) \times \mathcal{L} \\
\models' & \subseteq & (A \times \mathcal{D}_\omega C) \times \mathcal{L}' \\
\models'' & \subseteq & \mathcal{D}_\omega C \times \mathcal{L}''
\end{array}
$$

are defined by:

$$
\begin{array}{rcl}
X \models \Box\psi & \textit{iff} & \forall\, x \in X.\, x \models' \psi \\
\langle b, \mu\rangle \models' (a, \xi) & \textit{iff} & b = a \;\; \textit{and} \;\; \mu \models'' \xi \\
\mu \models'' \Diamond_p\varphi & \textit{iff} & \mu[\, [\![\varphi]\!]_\gamma \,] \geq p
\end{array}
$$

with $[\![\varphi]\!]_\gamma ::= \{\, c \in C \mid c \models_\gamma \varphi \,\}$.

7 Related Work

In [14], modal logics that capture various kinds of observational equivalence relations between (the states of) coalgebras were studied. The definition of such logics ultimately amounts to defining expressive logics for coalgebras of endofunctors of form $\mathcal{P}_\kappa \circ \mathsf{O}$, with the functor $\mathsf{O} : \mathsf{Set} \to \mathsf{Set}$ being used to define

[14] No bound on the cardinality of subsets is required in [15].

[15] Strictly speaking, propositional connectives are also allowed in the first component of formulae of form (a, ξ), but their presence does not add any additional expressive power.

a particular notion of observational equivalence. In defining an expressive logic for $\mathcal{P}_\kappa \circ \mathsf{O}$-coalgebras, a two-sorted language was used in [14] to interleave the $\mathcal{P}_\kappa$-structure with the O-structure at the level of formulae. The two steps used in that definition can be captured by two language constructors $\mathsf{S}_{\mathcal{P}_\kappa}$ and S_O (with $\mathsf{S}_{\mathcal{P}_\kappa}$ being as in Example 10), whereas the semantics of the language considered in [14] can be obtained by combining the $\mathcal{P}_\kappa$-semantics $\mathcal{F}_{\mathcal{P}_\kappa}$ for $\mathsf{S}_{\mathcal{P}_\kappa}$ of Example 19 with a suitable O-semantics $\mathcal{F}_\mathsf{O}$ for S_O. Then, the logic derived using our compositional approach coincides with the logic defined in [14].

As mentioned in the introduction, an extension of the compositional techniques presented here to the derivation of proof systems is described in [7]. There, a notion of *proof system constructor* is used to capture one inductive step in the definition of a proof system for a language for coalgebras. The soundness (completeness) of the induced proof system is subsequently shown to arise as a consequence of a soundness (completeness) condition on the underlying proof system constructor. By formalising the relationship between a language constructor and the induced language, the results in [7] also substantiate our claims in Examples 38, 39 and 40, that, in those particular cases, the induced languages for T-coalgebras coincide with the original languages.

8 Conclusions

We have presented a compositional approach to defining expressive logics for coalgebras of endofunctors on Set. This approach was based on notions of language constructor, associated semantics, and expressiveness of such a semantics. We have also shown that each of these notions can be treated in a modular fashion, and that combining such notions for different coalgebraic types yields expressive logics for coalgebras of increasingly complex types. An important application of these results was the derivation of expressive logics for a large class of probabilistic systems. In particular, we note that the presence of a combination of the powerset functor with the probability distribution functor in the coalgebraic modelling of probabilistic systems makes it difficult to propose suitable languages for coalgebras of such combinations, in the absence of systematic techniques. These difficulties are, however, overcome by the modular approach presented in this paper.

Acknowledgements

The final version of this paper has benefited greatly from discussions with Dirk Pattinson, as well as from the detailed comments and suggestions received from two anonymous referees.

References

[1] J. Adámek and V. Koubek. On the greatest fixed point of a set functor. *Theoretical Computer Science*, 150:57–75, 1995.

[2] M. Barr. Algebraically compact functors. *Journal of Pure and Applied Algebra*, 82(3):211–231, 1992.

[3] F. Bartels, A. Sokolova, and E. de Vink. A hierarchy of probabilistic system types. In H.P. Gumm, editor, *Coalgebraic Methods in Computer Science*, volume 82.1 of *Electronic Notes in Theoretical Computer Science*, 2003.

[4] F. Borceux. *Handbook of Categorical Algebra*, volume I. Cambridge University Press, Cambridge, 1994.

[5] F. Borceux. *Handbook of Categorical Algebra*, volume II. Cambridge University Press, Cambridge, 1994.

[6] C. Cîrstea. On expressivity and compositionality in logics for coalgebras. In H.P. Gumm, editor, *Coalgebraic Methods in Computer Science*, volume 82.1 of *Electronic Notes in Theoretical Computer Science*, 2003.

[7] C. Cîrstea and D. Pattinson. Modular construction of modal logics. Forthcoming.

[8] E.P. de Vink and J.J.M.M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221:271–293, 1999.

[9] B. Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *Theoretical Informatics and Applications*, 35:31–59, 2001.

[10] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[11] L.S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999.

[12] D. Pattinson. Expressive logics for coalgebras via terminal sequence induction. Technical report, LMU München, 2002.

[13] M. Rößiger. Coalgebras and modal logic. In H. Reichel, editor, *Coalgebraic Methods in Computer Science*, volume 33 of *Electronic Notes in Theoretical Computer Science*, 2000.

[14] J. Rothe and D. Mašulović. Modal logics for observational equivalences. In H.P. Gumm, editor, *Coalgebraic Methods in Computer Science*, volume 82.1 of *Electronic Notes in Theoretical Computer Science*, 2003.

[15] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[16] J. Worrell. On the final sequence of an accessible set functor. To appear in Theoretical Computer Science.