How to Write a Coequation

Fredrik Dahlqvist
Department of Computer Science, University College London, UK

Todd Schmid
Department of Computer Science, University College London, UK

Abstract

There is a large amount of literature on the topic of covarieties, coequations and coequational specifications, dating back to the early seventies. Nevertheless, coequations have not (yet) emerged as an everyday practical specification formalism for computer scientists. In this review paper, we argue that this is partly due to the multitude of syntaxes for *writing down* coequations, which seems to have led to some confusion about what coequations are and what they are for. By surveying the literature, we identify four types of syntaxes: *coequations-as-corelations*, *coequations-as-predicates*, *coequations-as-equations*, and *coequations-as-modal-formulas*. We present each of these in a tutorial fashion, relate them to each other, and discuss their respective uses.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Formal languages and automata theory; Theory of computation → Program specifications

Keywords and phrases Coalgebra, coequation, covariety

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.13

Category (Co)algebraic pearls

Funding *Fredrik Dahlqvist*: Leverhulme Project Grant 'Verification of machine learning algorithms'

Acknowledgements The authors are most grateful to Alexander Kurz for his services as history consultant. The responsibility for any mistake or mischaracterisation lie solely with the authors.

1 Introduction

Characterising algebras by the equations they satisfy is common practice. Equations are simple to write down: they consist of a pair of terms, or elements of an initial algebra. Equations are also simple to interpret: terms denote constructions, so an equation between two terms asserts that two constructions produce equivalent objects. The terms-as-constructions interpretation of equations is prevalent in programming language theory. A programming language is a syntax for denoting programs, so equations state equivalences between programs. Their importance can be seen in λ-calculus [15, 85, 84, 70], process algebra [35], Kleene algebra [86, 58, 24] and its extensions [59, 36, 56, 54, 91, 88], and related areas [94, 75].

A common thread, running through the many examples of equational reasoning in computer science, is that equations can be used to state *behavioural* equivalences between programs. For the purposes of this review, behaviours are what are obtained from dualizing, in the category theoretic sense, the concept of term. That is, a behaviour is an element of a final *coalgebra*. The dual study to algebra, coalgebra, constitutes a whole subfield of computer science dedicated to state-based dynamical systems, the sort of systems that exhibit behaviours [79, 43, 52]. Dualizing algebra not only takes terms to behaviours, but also equations to *coequations*, the main focus of this article.

Broadly, a coequation is a constraint on the dynamics of a state-based system. A coalgebra satisfies a coequation if its dynamics operate within the constraint. This situation is familiar to those working in automata theory, since deterministic automata are examples

9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021).
Editors: Fabio Gadducci and Alexandra Silva; Article No. 13; pp. 13:1–13:25
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of state-based dynamical systems. For a fixed alphabet A, any set of languages $\mathcal{L} \subseteq 2^{A^*}$ determines a coequation satisfied by those automata that only accept languages in $\mathcal{L}$. We call these coequations *behavioural*, as they consist of a set of states in the final automaton $2^{A^*} \to 2 \times (2^{A^*})^A$. Not all coequations are behavioural: following [93, 21] in viewing Kripke frames as coalgebraic dynamical systems for the powerset functor, modal formulas provide illustrative examples of nonbehavioural coequations. For example, reflexivity is a modally definable constraint on the dynamics of frames (witnessed by the modal formula $\Box p \to p$), despite the following two Kripke frames being behaviourally indistinguishable.

$$\circlearrowleft \bullet \qquad\qquad \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots$$

Rather, reflexivity is a coequation which requires two *colours* to be stated, p and $\neg p$. The concept of colour (or label) is key to moving beyond purely behavioural specifications, and can be understood as the formal dual to the notion of variable in algebra. Just like the commutativity of a binary operation requires two variables to be stated, the reflexivity of a Kripke frame requires two colours.

Coequations have arguably not seen much use by computer scientists, in spite of a large body of theoretical results. We postulate that one of the main reasons for this is that there is no one universally accepted way to *write down a coequation.* It is not hard to see why: while an equation relates two, finite, tree-structures which can be written-down unambiguously in one dimension with the use of brackets, there is no universal syntax for describing *constraints* on structures which are often *inherently infinitary.* In fact, a variety of syntaxes for writing down coequations have been proposed in the literature, leading to a certain ambiguity surrounding the term coequation, especially since some of them are less expressive than others. This being said, it could equally be argued that coequations are used extensively, if unknowingly, by computer scientists, in the shape of modal logics [18]. For example, languages like Linear Temporal Logic [92, 38, 37] and Computation Tree Logic [34, 22] are efficient syntaxes for specifying coequations.

The purpose of this paper is three-fold. The first is to survey and organise the literature on coequations. The second is to act as a tutorial on coequations and coequational specification. We assume basic knowledge of category theory and focus on Set-coalgebras. Finally, we aim to present a systematic account of what the various notions of coequations are, how they are related to one another, and what role they have to play in theoretical computer science.

The paper is structured as follows. We start with a review of the literature on coequations in § 2, highlighting a number of formalisms for defining and specifying coequations. We group these approaches into four paradigms, which we examine in detail. First, in § 3, we present a notion of coequation which dualizes exactly the notion of equation in Universal Algebra, and which we call *coequation-as-corelation.* Second, we present the view that a coequation is a predicate on a cofree coalgebra. We call this notion *coequation-as-predicate* and discuss it in detail in § 4. Third, we discuss *coequations-as-equations* in § 5 and relate them to coequations-as-corelations. The last paradigm we explore is that of coequations as modal formulas in § 6. Finally, we conclude in § 7 with some thoughts on the uses of each formalism and some recent appearances of coequations in computer science.

2 A brief history of coequations

As far as we are aware, the first mention of the word "coequation"– or more precisely "coequational" – dates back a series of papers by Davis [29, 30, 31, 32, 33] starting in 1970 and focusing on finding examples of comonadic/cotripleable categories. The earliest work that deals with covarieties of coalgebras as we now understand them, seems to be the 1985 paper [71], which presents a category-theoretic account of a dual to Birkhoff's HSP theorem. This work focuses on coalgebras for polynomial functors on the category Set. It describes an unusual approach to dualising Birkhoff's HSP theorem which reduces the problem to the ordinary version of the theorem by turning every coalgebra $X \to FX$ into an algebra $2^{FX} \to 2^X$, and by introducing an infinitary equational logic extending that of complete atomic boolean algebras to define varieties of such algebras. The idea of establishing a bridge between coequations and equations was explored again in [13] and [83] where conditions for a full duality between equations and coequations are given.

A few years after [71], work on the notion of *terminal coalgebra* in [1, 2, 16] laid the ground for [47, 76, 50], which proposed coalgebras as a semantic framework to formalise behaviours in object-oriented programming and infinite data structures such as streams and trees. From the onset, the aim of this line of research was to syntactically specify classes of behaviours and, although the terms "covariety" and "coequation" do not appear in *op.cit.*, a lot of the questions which we will explore in this paper can already be found in Hensel and Reichel's [47] and Jacobs' [50]. Both approaches propose *equational specifications* of coalgebras for polynomial functors, based on the signature of the functor. For example, the equation $\mathsf{head}(\mathsf{tail}(\mathsf{x})) = \mathsf{head}(\mathsf{x})$, where the "destructor" signature $\mathsf{head}, \mathsf{tail}$ can be read off the functor $F(X) = X \times A$, characterises constant streams in the terminal F-coalgebra A^ω. Such equations are called *state equations* in [47] since they must hold at every state, and [50] gives a concrete construction of the class of behaviours satisfying such an equation via the notion of "mongruence" (a terminology which mercifully has not caught on). This kind of coequational specification via equations, which we refer to as *coequations-as-equations*, is also used in Cîrstea's 1999 [20], which is the first full paper to use the term "coequation" in the sense we understand today. Roşu's [78] from 2001 follows the same approach of *coequations-as-equations*. While intuitive, this way of "writing coequations" is limited to endofunctors of a specific shape. Another, more powerful, way of writing coequations-as-equations was developed by Kurz and Rosickỳ in [68] using an equality relation between terms built from a signature, but with different notions of (co)operation and term. In this framework, every covariety over Set can be presented in a coequation-as-equation format.

Jacobs' [50] pioneered this specification format but also asked the following questions, which motivated a lot of the subsequent research on the topic: 1) Can a sound and complete logic to reason about coalgebras be devised? 2) Can a version of Birkhoff's theorems be proved for suitable classes of coalgebras? The first step towards answering these question was taken by Rutten's influential 1996 technical report [79, 80], which introduces the notion of "colours" of a cofree coalgebra, the concept of *covariety*, and the first of many dual versions of Birkhoff's HSP theorem. The term "coequation" does not appear in [79, 80], but it is worth noting that a concrete specification of a covariety is given by a *subcoalgebra of a cofree coalgebra*, in contrast to the equational presentation of [47, 50, 20, 78]. We refer to this type of coequational specification as *coequations-as-predicates*.

An important moment in the history of coequations was the first Workshop on Coalgebraic Methods in Computer Science (CMCS), organised by Jacobs, Moss, Reichel and Rutten and held in Lisbon in 1998. Several papers on covarieties were presented [45, 64, 77], and the next few years saw an explosion of research in this area. In retrospect, CMCS 1998 provided much of the momentum behind the subsequent blossoming of this new field of research.

In their 1998 CMCS paper [45], Gumm and Schröder pick up the study of covarieties defined via a subcoalgebra in [79]. They isolate precisely which subcoalgebras define a covariety and describe the closure properties of covarieties closed under bisimulation, which they call *complete covarieties*. It was subsequently shown in [11, 49] that these covarieties, more aptly called *behavioural covarieties*, are precisely those which can be described by a coequation-as-predicate over one colour, that is to say by a subcoalgebra of the terminal coalgebra. Coequations as subcoalgebras of a cofree coalgebras (or more abstractly as regular monomorphisms with cofree codomain) and the covarieties they define are also discussed in detail in [6, 7] where a dual to Birkhoff's HSP theorem is given.

Hughes' 2001 thesis [49] presents a very detailed abstract account of the *coequations-as-predicates* perspective. A coequation is no longer required to be defined by a sub*coalgebra*, but can simply be a sub*set* of a cofree coalgebra [49, §3.6.3]. Closure operators defined and studied in [49, 48, 51] connect these two flavours of the *coequations-as-predicates* paradigm, by constructing the (invariant) sub*coalgebra* generated by a sub*set* of behaviours. More abstractly, [49] also considers a coequation as a subcoalgebra of a regular injective coalgebra (e.g. a cofree coalgebra). This additional abstraction dualizes the description of equations as quotients of regular projective algebras due to [14], but introduces subtle differences on the closure properties of covarieties which are discussed by Goldblatt in [39, 23]. In this paper, we only consider subcoalgebras of cofree coalgebras or subsets of their carriers.

A third version of the *coequations-as-predicates* paradigm was proposed by Gumm in [44], where a coequation is a single pattern (i.e. element of a cofree coalgebra), but a pattern that *must be avoided.* In other words, this is a *coequation-as-predicate* defined by the complement of a singleton, i.e. understood as a *pattern avoidance constraint.* This fruitful idea was the source of many interesting examples in [7], and the basis for coequational logics in [4, 89, 90]. These logics are based on two observations which can already be found in [44], namely that if there exists a state x witnessing a pattern f, then any "successor pattern" of f must be witnessed by some state y (namely one of the successors of x). Similarly, if there exists a state x witnessing a pattern f, there must exist a recolouring/relabelling of this state which witnesses a similar relabelling of the pattern f. Adamek [4] shows that these two observations are enough to define a sound and complete coequational system in which new coequations (avoidance patterns) can be deduced from known ones. This logic is most natural for polynomial functors, but can also be made to work for very large class of accessible functors through the notion of functor presentation [4, 89, 90].

Finally, we mention generalisations of *coequations-as-predicates* to the case where cofree coalgebras do not exist. Adámek and Porst generalise coequations-as-predicates by considering regular monomorphisms into *any* element of the cofree coalgebra chain, whether it stabilises or not [7]. Kurz and Rosickỳ in [68] describe the notion of *implicit operations* which permits an equivalent notion of coequation-as-predicate for functors which have no cofree coalgebras. Finally, Adámek describes a comprehensive solution to this problem by considering generalised coequations-as-predicates as subchains of the entire cofree-coalgebra chain in [3].

A related but different notion of coequation was proposed in 2000 by Wolter [95] and Kurz [65], systematically dualizing the picture from categorical Universal Algebra. Since a set of equations can be understood as a relation between terms in a free algebra (categorically a span), Wolter proposes that a coequation should be seen as a *corelation* on the carrier of a cofree coalgebra over some set of colours (categorically, a cospan). Similarly, Kurz proposes to consider a *cocongruence* on the cofree coalgebra. As in the *coequation-as-predicate* paradigm, these two approaches reflect the fact that one may consider a coequation as a structure on the *carrier* of a cofree coalgebra (a corelation), or on the cofree coalgebra itself (a cocongruence). We will refer to this approach as *coequations-as-corelations.* This approach neatly dualizes the well-known theory of equations, but the notion of corelation is not very intuitive, as pointed out by Hughes [49]. Nevertheless, we hope to provide some intuition in § 3 and § 5.

At the same period Kurz also proposed *modal logic* as a language for specifying covarieties [63, 65, 66, 67]. This is our last paradigm for coequations: *coequations-as-modal-formulas*. Following our discussion in the introduction, we know that for finitely branching Kripke frames there exists a *final* Kripke frame coloured by (sets of) propositional variables, and any modal formula ϕ selects the states in this Kripke frame in which ϕ is valid, i.e. defines a predicate on a cofree coalgebra. A modal formula can thus be seen as a syntax for *coequations-as-predicates*. However, these are very particular predicates: they have a simple and intuitive syntax, access to a countable set of colours (the propositional variables), and to 1-step ahead colours (through modalities). This idea can easily be extended to modal logics for polynomial functors [66]. However, the idea of modal logic as a specification language for covarieties found in [63, 65, 66, 67] was in some way too prescient: the appropriate extension of modal logic – *coalgebraic modal logic* – was still in its infancy. Moss' logic [73] had only just been published, and neither the predicate lifting formalism of Pattinson [74] nor the abstract formalism of Kupke, Kurz, and Pattinson [60, 61, 53] had been developed. As a consequence, Kurz's insight of *coequations-as-modal-formulas* was only worked out for standard modal logics [66]. His abstract notion of *modal predicate* [65] – where the term "modal" is meant as "invariant under bisimulation" – is not based on a particular syntax, but is defined as a monomorphism into a cofree coalgebra, i.e. as a coequation-as-predicate.

3 Coequations-as-corelations

Coequations-as-corelations is the notion of coequation which most faithfully dualizes the notion of equation from Universal Algebra. It is not the simplest approach, but it exposes the underlying machinery in its entirety. Syntactically, it is a formalism that resembles equations because it uses a *pairs of expressions*. However, whilst an equation between a pair of expressions *forces* an equality to be witnessed via a quotient, a *coequation-as-corelation* involves a pair of expressions which *selects* an "existing" equality via an equaliser. Much of the material in this section can be found in [95, 65, 11, 48, 49, 25]. We present the classical picture from Universal Algebra in § 3.1, and then dualize it in § 3.2.

3.1 Equations, relations and varieties of algebras

We will not go to the level of generality of [11, 49, 48] but instead focus on algebras for Set-endofunctors. Let $T : \mathsf{Set} \to \mathsf{Set}$ be an endofunctor and let $\mathsf{Alg}(T)$ denote the category of T-algebras and T-algebra morphisms. There exists an obvious forgetful functor $U_T : \mathsf{Alg}(T) \to \mathsf{Set}$ which keeps the carrier and forgets the algebraic structure. A functor T is called a *varietor* [9] if this functor has a left-adjoint $F_T : \mathsf{Set} \to \mathsf{Alg}(T)$ which builds free T-algebras over any given set of variables. We will drop the subscripts and simply write $F \dashv U$ if this causes no ambiguity. It follows from the adjunction that any map $h : X \to U(A, \alpha)$ can be freely extended to a T-algebra morphism $\hat{h} : FX \to (A, \alpha)$ explicitly constructed as $\hat{h} \triangleq \varepsilon^T_{(A,\alpha)} \circ Fh$, where ε^T is the counit of the adjunction.

For any varietor T we define a *set of T-equations over a set of variables X* is a pair of arrows $e_1, e_2 : E \rightrightarrows UFX$. A set of equations is thus represented as a *span*, the categorical embodiment of the notion of relation.

A T-algebra (A, α) *satisfies a set of equations* $e_1, e_2 : E \rightrightarrows UFX$ if for all *valuations* $v : X \to U(A, \alpha)$, $U\hat{v} \circ e_1 = U\hat{v} \circ e_2$, i.e. if the map $\hat{v}$ which recursively computes the interpretation in (A, α) of formal terms from FX, returns the same output for the left- and right-hand-side of each equation in E. This can be rephrased as a universal property in $\mathsf{Alg}(T)$ by saying that (A, α) satisfies a set of equations $e_1, e_2 : E \rightrightarrows UFX$ if any T-algebra morphism $f : FX \to (A, \alpha)$ factors uniquely through the coequalizer q of $\hat{e}_1, \hat{e}_2$.

$$\begin{tikzcd}
FE \arrow[r, shift left, "\hat{e}_1"] \arrow[r, shift right, "\hat{e}_2"'] & FX \arrow[r, two heads, "q"] \arrow[d, "f"] & (Q, \nu) \arrow[dl, dashed] \\
& (A, \alpha) &
\end{tikzcd} \tag{1}$$

Since any morphism $f : FX \to (A, \alpha)$ is of the shape $f = \hat{v}$ for some $v : X \to U(A, \alpha)$, we recover the standard notion of equation satisfaction. An object (A, α) with the universal property in (1) is said to be *orthogonal* to $q : FX \twoheadrightarrow (Q, \nu)$, written $q \perp (M, \alpha)$.

The *variety of T-algebras defined by the set of T-equations $e_1, e_2 : E \rightrightarrows UFX$* is defined as the class of all T-algebras which are orthogonal to the coequalizer of the adjoint morphisms $\hat{e}_1, \hat{e}_2 : FE \rightrightarrows FX$, notation $q^\perp$. Equivalently, a variety of T-algebras is a class of T-algebras orthogonal to a regular epi $q : FX \twoheadrightarrow Q$,[1] a definition which dates back to [14]. With this terminology in place we state Birkhoff's famous HSP theorem.

▶ **Theorem 1** ([17, 87, 49, 6, 7]). *Let $T : \mathsf{Set} \to \mathsf{Set}$ be a varietor. A class of T-algebras is a variety iff it is closed under Homomorphic images (H), Subalgebras (S), and Products (P).*

▶ **Example 2.** Recall that a monoid is a set M equipped with a binary operation $* : M \times M \to M$ and a constant $e \in M$ satisfying the three *equations*: $x * (y * z) = (x * y) * z$, $e * x = x$, and $x * e = x$. Every monoid is an algebra for the functor $\Sigma M = M \times M + 1$, and the functor Σ is a varietor: F_Σ builds the set of all formal terms constructed from the signature and a set of variables (e.g. $\{x, y, z\}$), and equips it with the trivial Σ-algebra structure taking the unit to be the *term* e, and the product of two terms s, t to be the *term* $s * t$. The equations of the theory of monoids can be described as the pair of maps $e_1, e_2 : 3 \rightrightarrows UF\{x, y, z\}$, where $3 \triangleq \{0, 1, 2\}$ and for $0 \leqslant i \leqslant 2$, $e_1(i)$ (resp. $e_2(i)$) picks the left-hand-side (resp. right-hand-side) of the i^{th} equation above. A monoid is a Σ-algebra in the variety defined by the coequalizer of the adjoint morphisms $\hat{e}_1, \hat{e}_2 : F3 \rightrightarrows F\{x, y, z\}$ which homomorphically sends formal terms on 3 to terms on $\{x, y, z\}$, for example $\hat{e}_1(1 * 2) = (e * x) * (x * e)$. The relation defined by the span $U\hat{e}_1, U\hat{e}_2 : UF3 \rightrightarrows UF\{x, y, z\}$ is thus closed under the rule

$$\frac{s_1 = t_1 \quad s_2 = t_2}{s_1 * s_2 = t_1 * t_2}\ \textbf{p-cong}$$

For example, $(x * e) * (e * x) = x * x$ is an equation belonging to this relation. Such a relation on terms is called a *pre-congruence* in [49].

The rule **p-cong** generalizes easily to all polynomial functors, but it is not obvious how it should be adapted to the general case. Therefore, we simply say that the relation defined by a span on UFX is a *pre-congruence* if it is of the shape $U\hat{e}_1, U\hat{e}_2 : UFE \rightrightarrows UFX$.

It is important to note that the quotient (Q, ν) in (1) is *not* a member of the variety. In Example 2, $y * e$ and y belong to different equivalence classes in Q since the quotient q only needs to identify $x * e$ and x. The interpretations of $y * e$ and y are equal in all objects belonging to the variety of monoids because of the universal quantification over the morphism f in (1) which takes care of all substitutions. In order to build a quotient that *does* belong to the variety we need more equations than those in the set UFE. It is well known that

[1] By taking $U \ker(q) \rightrightarrows UFX$ as the set of equation and using the fact that U is monadic [5, 20.56], it can be shown that we recover the quotient q as the coequalizer of the lifted equations.

equational reasoning also adheres to the following rules:

$$\frac{}{t=t}\ \textbf{ref} \qquad \frac{t=s}{s=t}\ \textbf{sym} \qquad \frac{s=t \quad t=u}{s=u}\ \textbf{trans}$$

A relation on UFX which is closed under **ref**, **sym**, and **trans** is called an *equivalence relation*, and a pre-congruence which is also an equivalence relation is called a *congruence*. It is easy to turn the relation defined by (1) into a congruence by taking the *kernel pair* of the coequalizer q, i.e. by moving to the exact sequence $\ker(q) \rightrightarrows FX \twoheadrightarrow (Q,\nu)$. Since any coequalizer is also the coequalizer of its kernel pair, q remains the coequalizer, and thus the variety it defines remains the same. We have now increased the collection of derivable equations. Following Example 2, the congruence $\ker(q) \rightrightarrows F\{x,y,z\}$ contains the equation $e*x = x*e$, for example, which requires **sym** and **trans** to derive.

As the reader will have guessed, we need to add substitution instances. Starting from the pre-congruence of (1), this can be done categorically [25, §1.4] by considering the coequalizer

$$\begin{tikzcd} \coprod_{v\in V} FE \arrow[r, shift left, "{[\hat{v}\circ\hat{e}_1]_{v\in V}}"] \arrow[r, shift right, "{[\hat{v}\circ\hat{e}_2]_{v\in V}}"'] & FX \arrow[r, two heads, "q'"] & (Q',\nu') \end{tikzcd} \tag{2}$$

where V is the set of all substitutions $V = \{v : X \to UFX\}$. It is not difficult to see that (Q',ν') now *does* belong to the variety defined by $q' : FX \twoheadrightarrow (Q',\nu')$, i.e. $q' \perp (Q',\nu')$.

Since F is a left-adjoint, $\coprod_{v\in V} FE \simeq F(\coprod_{v\in V} E)$, i.e. (2) involves the free T-algebra generated by all substitution instances of the axioms UFE. The relation defined by the span $U\coprod_{v\in V} FE \rightrightarrows UFX$ is a pre-congruence closed under the substitution rule

$$\frac{s=t \qquad v\in V}{\hat{v}(s)=\hat{v}(t)}\ \textbf{subst}$$

Applying (2) to Example 2, we get that $y*e=y$ now belongs to the stock of equations. In fact, it appears several times, since any substitution mapping x to y will produce it.

Following [49] we say that a set of equations $e_1,e_2 : E \rightrightarrows UFX$ is *stable* if it is closed under substitutions in the sense that for any $v\in V$ there exists a (necessarily unique) map $\tilde{v} : E \to E$ such that $e_i\circ\tilde{v} = U\hat{v}\circ e_i, i=1,2$. Taking the kernel pair

$$\begin{tikzcd} \ker(q') \arrow[r, shift left] \arrow[r, shift right] & FX \arrow[r, two heads, "q'"] & (Q',\nu') \end{tikzcd} \tag{3}$$

constructs a stable set of equations $U\ker(q') \rightrightarrows UFX$ by construction [25, §1.4].

We have described three categorical constructions – lifting the equations, closing under (2), and taking the kernel pair of the coequalizer (3) – which, combined, turn a *set* of equations $e_1,e_2 : E \rightrightarrows UFX$ into an exact sequence $\ker(q') \rightrightarrows FX \twoheadrightarrow (Q',\nu')$ which defines a *stable congruence* $U\ker q' \rightrightarrows UFX$. We do not know if this purely categorical procedure produces the *smallest* stable congruence, and we do not know if the order in which the three steps are carried out matters. These questions are also raised in [48, §8], and as far as we could see, no simple categorical argument can answer them. What is clear however, is that this construction defines a quotient of the free T-algebra which belongs to the variety it defines.

▶ **Theorem 3** ([49] Thm 3.5.3)**.** *Let T be a varietor, let $e_1,e_2 : E \rightrightarrows UFX$ be a stable set of T-equations over X, and consider the coequalizer*

$$\begin{tikzcd} FE \arrow[r, shift left, "\hat{e}_1"] \arrow[r, shift right, "\hat{e}_2"'] & FX \arrow[r, two heads, "q"] & (Q,\nu). \end{tikzcd}$$

Then $q \perp (Q,\nu)$. Conversely, if $q \perp (Q,\nu)$, then $\ker(q)$ *is stable.*

We finish by stating Birkhoff's completeness theorem for equational reasoning. Given a collection $\mathbb{V}$ of T-algebras (e.g. a variety), define $\mathrm{Eq}(\mathbb{V})$ as the set of equations satisfied by every T-algebra in $\mathbb{V}$, i.e.

$$\mathrm{Eq}(\mathbb{V}) = \{e_1, e_2 : 1 \rightrightarrows UFX \mid \forall (M, \alpha) \in \mathbb{V}, \forall v : X \to M, U\hat{v} \circ e_1 = U\hat{v} \circ e_2\}.$$

▶ **Theorem 4** (Birkhoff's completeness theorem, e.g. [17, 87, 49]). *Let Σ be a polynomial functor, and let $E \rightrightarrows U_\Sigma F_\Sigma X$ be a set of equations. Then $E = \mathrm{Eq}(\mathbb{V})$ for some variety $\mathbb{V}$ iff E is closed under* ***p-cong**, **subst**, **ref**, **sym** and **trans** .*

3.2 Coequations, corelations and covarieties of coalgebras

Next, we dualize the concepts developed in §3.1. We denote by $\mathsf{CoAlg}(T)$ the category of T-coalgebras and T-coalgebra homomorphisms. A functor $T : \mathsf{Set} \to \mathsf{Set}$ is called a *covarietor* [6] if the forgetful functor $U_T : \mathsf{CoAlg}(T) \to \mathsf{Set}$ has a *right* adjoint $C_T : \mathsf{Set} \to \mathsf{CoAlg}(T)$, called the *cofree functor*. For any set X, the coalgebra C_TX is called the *cofree coalgebra over the set of colours X*, or the *cofree coalgebra in X colours*. We omit subscripts if there is no risk of confusion. Intuitively, C_TX is the collection of *X-patterns*, T-processes (histories of states in a T-transition system) whose states are labelled by elements of X.

Given a covarietor T we dualize the notion of equation by defining a *T-coequation in X colours* [95] to be a *cospan* $c_1, c_2 : UCX \rightrightarrows 2$. The role of $2 \triangleq \{0, 1\}$ is dual to the role of 1 in the definition of a T-equation, since 2 is a cogenerator in Set, whilst 1 is a generator. Following [95], we define a *T-coequational specification S* as a pair of maps $c_1, c_2 : UCX \rightrightarrows S$. This concept dualizes the notion of a set of T-equations, and whilst a set of T-equations is equivalent to a relation on UFX, a coequational specification defines a *corelation*, i.e. a map $UCX + UCX \twoheadrightarrow S$. One should think of a corelation on UCX as two different classification schemes – in the case of a coequation, two binary classification schemes, accepting or rejecting behaviours – used to select the behaviours/patterns that they cannot distinguish.

The dual to the notions of valuation and interpretation/substitution are the notions of *colouring* and *recolouring map*. Given a T-coalgebra (V, γ), a function $k : V \to Y$ is called a *Y-colouring map*, as it labels the states of the coalgebra with the colours of Y. Given such a colouring map, we call its cofree extension $\hat{k} : (V, \gamma) \to CY$ a *recolouring map*, $\hat{k} \triangleq C_Tk \circ \eta^T_{(V,\gamma)}$. Starting at a state $v \in V$, this map follows the history of the T-transition system (V, γ), reads the colour(s) of the successor state(s) at each time step, and uses this information to construct the T-transition system of observed colours. The original T-history is typically infinite, and therefore so is the T-history of its colours. Thus, $\hat{k}$ is a map which typically processes an entire infinitary structure in one go.

Colouring maps of the shape $k : UCX \to Y$ are important to understanding coequations. Imagine an omniscient being that can examine the entire (possibly infinite) history of an X-pattern in UCX and then classify it according to a rule of her choosing with a set of labels Y. This is a colouring map on UCX. In particular, for a colouring map $k : UCX \to X$, the omniscient being can examine the entire X-labelling history of a T-process and aggregate this information into a single X-label. Every cofree coalgebra comes with such a canonical colouring map $\varepsilon^T_X : UCX \to X$ provided by the counit ε^T of the adjunction $U \dashv C$, which returns the colour of the initial state. Given a colouring map $k : UCX \to Y$, the associated recolouring map $\hat{k} : CX \to CY$ can be understood as the process by which our omniscient being can follow an entire T-process and, at each time-step, classify the *remaining history* of the T-process according to the colouring map k. In this way the omniscient being can build the *labelling history* of the entire process in one single evaluation.

The *covariety defined by a T-coequational specification $c_1, c_2 : UCX \rightrightarrows S$ over a set (of colours) X*, is the class of coalgebras (V, γ) such that for every colouring map $k : V \to X$, there is a unique coalgebra morphism from (V, γ) to the *equaliser* (H, ξ) of $\hat{c}_1, \hat{c}_2$ such that

$$\begin{tikzcd}
(H,\xi) \arrow[r, tail, "m"] & CX \arrow[r, shift left, "\hat{c}_1"] \arrow[r, shift right, "\hat{c}_2"'] & CS \\
& (V,\gamma) \arrow[ul, dashed] \arrow[u, "\hat{k}"'] &
\end{tikzcd} \tag{4}$$

commutes.[2] The coalgebra (V, γ) is said to be co-orthogonal to the regular mono m, written $m \top (V, \gamma)$, and the covariety defined by (4) can be described as the collection of coalgebras $m^{\top}$ which are co-orthogonal to m. With this definition, we can state the dual to Theorem 1:

▶ **Theorem 5** (co-Birkhoff (HSC) theorem, e.g. [65, 49, 6, 7])**.** *Let T : Set $\to$ Set be a covarietor. A class of T-coalgebras is a covariety iff it is closed under Homomorphic images (H), Subcoalgebras (S) and Coproducts (C).*

Dual to the case of varieties, closure properties of the corelation $c_1, c_2 : UCX \rightrightarrows S$ can ensure that we obtain an equaliser which belongs to the covariety. Most of the literature focuses on closure properties on the subobject side of (4), e.g. the notion of mongruence [50], the (modal) closure operators of [49, 48], and the notion of invariant subcoalgebra of [45]. Since we dedicate §4 to this perspective, we follow [95, 65] and focus on the quotient.

Merging the nomenclatures of [49] and [65], we call the cospan $\hat{c}_1, \hat{c}_2 : CX \rightrightarrows CS$ a *pre-cocongruence*, the notion dual to a pre-congruence. Thus a pre-cocongruence is a corelation that has a coalgebra structure compatible with that of CX, i.e. which is closed under taking successors.[3] Following [95], we will say that a corelation $c_1, c_2 : UCX \rightrightarrows S$ is *coreflexive* if there exists a map $s : S \to UCX$ such that $s \circ [c_1, c_2] = [\mathrm{id}_{UCX}, \mathrm{id}_{UCX}]$, i.e. if it is only allowed to identify a behaviour $(1, t)$ in the first component of the coproduct with a behaviour $(2, s)$ in the second if $t = s$. There is also a notion of cosymmetric, cotransitive, and of coequivalence corelation, but it turns out that in Set these are implied by being coreflexive [95]. Following our earlier definition of pre-congruence and congruence, we will say that a pre-cocongruence is a cocongruence if it is coreflexive. It is easy to turn any corelation into a coreflexive corelation, it suffices to consider the *cokernel of its equaliser*. Finally, dual to the notion of a stable set of equations, we will say that a corelation $c_1, c_2 : UCX \rightrightarrows S$ is *invariant* if for any colouring map $k : UCX \to X$ there exists a (necessarily unique) morphism $\tilde{k} : S \to S$ such that $c_i \circ \tilde{k} = U\hat{k} \circ c_i, i = 1, 2$. By dualizing (2)-(3) we can turn any corelation into an invariant corelation by first considering the equalizer of the cospan

$$\begin{tikzcd}
(H',\xi') \arrow[r, tail, "m'"] & CX \arrow[r, shift left, "\langle \hat{c}_1 \circ \hat{k} \rangle_{k \in K}"] \arrow[r, shift right, "\langle \hat{c}_2 \circ \hat{k} \rangle_{k \in K}"'] & \prod_{k \in K} CS
\end{tikzcd} \tag{5}$$

where $K = \{k : UCX \to X\}$ is the set of X-colouring maps. Since C is right-adjoint, it preserves products and $\prod_k CS \simeq C \prod_k S$. Thus, we are considering as corelation a pair of maps which can perform two "S-classifications" of an X-pattern *and all its X-recolourings*, simultaneously. Clearly, $m' \top (H', \xi')$, so (H', ξ') belongs to the covariety it defines.

[2] For any Set-endofunctor the category CoAlg(T) always has equalisers [43, 5.1]

[3] This is what Kurz calls a cocongruence in [65].

By taking the cokernel pair of the equalizer m' above $(H', \xi') \rightarrowtail CX \rightrightarrows \mathrm{coker}(m')$ we get a corelation which is invariant by construction. In fact, we get an *invariant cocongruence*, which are to cofree coalgebras what stable congruences are to free algebras. We can now state the dual of Theorem 3:

▶ **Theorem 6.** *Let T be a covarietor, let $c_1, c_2 : UCX \rightrightarrows S$ be an invariant T-coequational specification, and consider the equalizer*

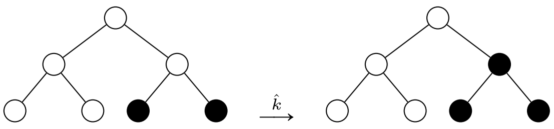
$$\begin{tikzcd}(H, \xi) \arrow[r, tail, "m"] & CX \arrow[r, shift left, "\hat{c}_2"] \arrow[r, shift right, "\hat{c}_1"'] & CS.\end{tikzcd}$$

Then $m \top (H, \xi)$. Conversely, if $m \top (H, \xi)$, then $\mathrm{coker}(m)$ *is invariant.*

▶ **Example 7.** Let us consider the same endofunctor as in Example 2, namely the functor $\Sigma X = X \times X + 1$. This functor is a covarietor and the cofree Σ-coalgebra $C_\Sigma X$ over X is the set of finite and infinite binary trees whose nodes are labelled by the elements of X [7]. Consider in particular the cofree Σ-coalgebra over a set of two colours, which we shall write as $\{b, w\}$ for "black" and "white". We now define the coequation $c_1, c_2 : UC\{b, w\} \rightrightarrows 2$

$$c_1(t) = \begin{cases} 1 & \text{if } \mathsf{LeftChild}(t) \text{ is labelled } b \\ 0 & \text{else} \end{cases} \qquad c_2(t) = \begin{cases} 1 & \text{if } \mathsf{RightChild}(t) \text{ is labelled } b \\ 0 & \text{else} \end{cases}$$

where $\mathsf{LeftChild}(t)$ being labelled b assumes that it exists, i.e. that t is not a leaf state, and similarly for $\mathsf{RightChild}(t)$. Recall that a *coequation-as-corelation* defines the set of behaviours which cannot be distinguished by the two classification schemes. The coequation above defines the covariety of finite and infinite binary trees with the property that if a state has left and a right children states, then they must be equal; i.e. the covariety of deterministic binary trees. To see this, consider the equalizer $m : (H, \xi) \to CX$ of $\hat{c}_1, \hat{c}_2$. It contains all binary trees such that left- and right-successors share a colour. Its cokernel defines the cocongruence on $CX + CX \to C\{b, w\}$, which only identifies $(1, s)$ and $(2, t)$ if $s = t$ belongs to H. This cocongruence is not invariant: the two copies in $CX + CX$ of the left-hand tree t_l below are identified by the corelation c_1, c_2 (and its coreflexive closure), but can be recoloured into two copies of the right-hand tree t_r, which will be kept distinct by the corelation.



By constructing the invariant closure of the corelation using the construction of (5), the two trees above become components in the tuple of all recolourings of t_l. Applying $\hat{c}_1$ and $\hat{c}_2$ component-wise to this tuple will yield two tuples which will disagree at the coordinate of t_r. In fact, the equalizer (H', ξ') of $\langle \hat{c}_1 \circ \hat{k} \rangle_{k \in K}, \langle \hat{c}_2 \circ \hat{k} \rangle_{k \in K} : C\{b, w\} \rightrightarrows \prod_k C2$ contains precisely the trees whose nodes at depth n all have the same colour, in other words the equalizer is isomorphic to $\{b, w\}^* \cup \{b, w\}^\omega$. From this it follows that given an arbitrary Σ-coalgebra (V, γ), if *any* recolouring map $\hat{k} : (V, \gamma) \to C\{b, w\}$ has to factor through $\{b, w\}^* \cup \{b, w\}^\omega$ it must be the case that $\pi_1(\gamma(x)) = \pi_2(\gamma(x))$ or $\gamma(x) \in 1$ for all $x \in V$.

4 Coequations-as-predicates

The coequations-as-predicates paradigm provides a picture of coequations that is very flexible with regard to how they can be written. In this paradigm, a coequation is a subset of a cofree coalgebra, so any method of describing subsets can be used. In practice, the elements of a cofree coalgebra carry some structure, for eg. a tree or a stream of numbers, allowing the user to describe coequations in terms of this structure.

In §4.1 and §4.2, we will see some examples of predicate coequations and their descriptions. We then give a brief account of Adámek and Schwencke's observation that there is an inherent logical structure to coequations in §4.3. Finally, in §4.4, we talk about the expressiveness of predicate coequations in general, and give a generalization of predicate coequations when there are no cofree coalgebras.

In this section, we entirely focus on coalgebras in Set. Many of the results can be generalized to coalgebras over other base categories, see for example [7] and the thesis [49].

4.1 Behavioural coequations

Fix a covarietor T on Set with forgetful-cofree adjunction $U \dashv C$, and let (Z, δ) denote the final coalgebra $C1$. Given two coalgebras $(V_1, \gamma_1), (V_2, \gamma_2)$, two states $v_1 \in V_1$ and $v_2 \in V_2$ are said to be *behaviourally equivalent* (e.g. [62]) if there is a third coalgebra (V', γ') and homomorphisms $h_i : V_i \to V'$ such that $h_1(v_1) = h_2(v_2)$. Behavioural equivalence is an equivalence relation: it is reflexive and symmetric, and since the category $\mathsf{CoAlg}(T)$ has pushouts, it is also transitive. Furthermore, since every coalgebra (V, γ) admits a unique coalgebra homomorphism $!_V : (V, \gamma) \to (Z, \delta)$, the state $!_V(v)$ of Z is a representative of the behavioural equivalence class of v for any state $v \in V$. This is the motivation for calling the states of (Z, δ) *behaviours* (for the functor T).

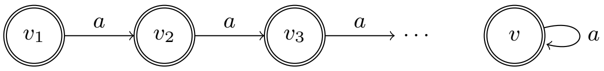
A *behavioural coequation* is a subset of Z. A coalgebra (V, γ) *satisfies* $W \subseteq Z$, written $(V, \gamma) \models W$, if $\mathrm{im}(!_V) \subseteq W$, i.e. a coalgebra satisfies a behavioural coequation if the coequation contains all of the behaviours exhibited by the coalgebra. A *behavioural covariety* is a class of coalgebras of the form $\mathsf{Cov}(W) = \{(V, \gamma) \mid (V, \gamma) \models W\}$ for some $W \subseteq Z$.

▶ **Example 8.** Coalgebras for the functor $T_{det} = 2 \times \mathrm{Id}^A$ are deterministic automata. The final deterministic automaton is $(2^{A^*}, \langle \epsilon?, \partial \rangle)$, where A^* is the set of empty or nonempty words in the alphabet A (here, ϵ is the empty word), and

$$\epsilon?(L) = \begin{cases} 1 & \text{if } \epsilon \in L \\ 0 & \text{otherwise} \end{cases} \qquad \partial(L)(a) = \{w \in A^* \mid aw \in L\}$$

for any $L \subseteq A^*$ and $a \in A$ [79, 19]. Recall that the set $\mathsf{Reg} \subseteq 2^{A^*}$ of *regular languages* is the smallest subset of 2^{A^*} closed under concatenation, iteration, and finite unions, and containing $\{\epsilon\}$ and $\{a\}$ for each $a \in A$. The class $\mathsf{Cov}(\mathsf{Reg})$ of deterministic automata that accept regular languages is a behavioural covariety.

By Kleene's theorem, a deterministic automaton satisfies Reg if and only if it is bisimilar to a locally finite automaton. There are many examples of deterministic automata that satisfy Reg but are not locally finite: The following automata are bisimilar and both accept the language a^*, but only one of them is locally finite.



The relationship between regular languages and finite automata in Example 8 is typical of behavioural coequations. Behavioural coequations constrain dynamics by constraining behaviour, and behaviour is preserved under many of the useful operations on coalgebras. In general, behavioural coequations carve nicely structured categories out of $\mathsf{CoAlg}(T)$. Following the co-Birkhoff result of Theorem 5, we will say that a class of coalgebras is a *structural covariety* if it is closed under homomorphic images, subcoalgebras and coproducts. Note that this concept makes sense whether T is a covarietor or not.

▶ **Proposition 9** (Rutten [79])**.** *For any* $W \subseteq Z$*,* $\mathsf{Cov}(W)$ *is a* structural covariety*.*

However, not every structural covariety is carved out of $\mathsf{CoAlg}(T)$ by a behavioural coequation. As we saw in Example 8, locally finite deterministic automata are not behaviourally specified: if they were, their defining coequation would be Reg. On the other hand, we will see in §4.2 that (under mild conditions) locally finite coalgebras form a structural covariety.

The mismatch between behavioural coequations and covarieties does not detract from the importance of behavioural constraints in the computer science literature. Behavioural coequations are particularly common in fields like automata theory and process algebra where specification languages play an important role.

▶ **Example 10.** Fix a set A of *atomic actions*, and consider the following BNF grammar

$$E ::= a \in A \mid x \in \mathrm{Var} \mid E + E \mid a(E) \mid \mu x.F \qquad F ::= a \in A \mid F + F \mid a(E) \mid \mu x.F$$

A *specification* is an expression $e \in E$ in which every variable $x \in \mathrm{Var}$ appears within the scope of a μx. The set of specifications can then be given the structure of a $\mathcal{P}_\omega(\{\checkmark\} + \mathrm{Id})^A$-coalgebra (Exp, ∂) using the GSOS law below. For any $a \in A$, $e, e_1, e_2, f \in \mathrm{Exp}$, infer

$$\frac{}{a \xrightarrow{a} \checkmark} \qquad \frac{}{a(e) \xrightarrow{a} e} \qquad \frac{e_1 + e_2 \xrightarrow{a} f}{e_i \xrightarrow{a} f} \qquad \frac{\mu x.f \xrightarrow{a} e}{f[\mu x.f/x] \xrightarrow{a} e}$$

Here, $e \xrightarrow{a} \xi$ means that $\xi \in \partial(e)(a)$. The functor $\mathcal{P}_\omega(\{\checkmark\} + \mathrm{Id})^A$ has a final coalgebra (Z, δ) consisting of A-decorated trees with transitions carrying labels from A and leaves carrying the label ✓ [96]. Every specification e gives rise to a unique behaviour $!_{\mathrm{Exp}}(e)$, and therefore also a tree. In analogy with regular languages (see Example 8), one might call the set of behaviours arising from specifications the *regular* coequation. A process satisfies the regular coequation if every of its states mimics the behaviour of a specification.

Many pairs of specifications give rise to identical behaviours: For example, $e + f$ and $f + e$ are behaviourally equivalent for any e and f, and so are $f[\mu x.f/x]$ and $\mu x.f$. Studying behavioural equivalences like these is a popular topic in process algebra [35].

The reader familliar with process algebra should note that in many cases, including Example 10, behavioural equivalence and *bisimilarity* coincide. For a general T, a bisimulation between T-coalgebras (V_1, γ_1) and (V_2, γ_2) consists of a coalgebra (R, ρ) and a pair of coalgebra homomorphisms $\pi_i : (R, \rho) \to (V_i, \gamma_i)$. Image factorisations exist in Set, so every bisimulation is equivalent to one in which $R \subseteq V_1 \times V_2$ and the homomorphisms π_1, π_2 are the projections of R onto the first and second components of R [79]. If two states $v_1 \in V_1, v_2 \in V_2$ are related by a bisimulation, we say that v_1 and v_2 are *bisimilar* and write $v_1 \leftrightarroweq v_2$. Important examples of bisimulations include graphs of homomorphisms: in fact, a function $V_1 \to V_2$ is a coalgebra homomorphism if and only if its graph is a bisimulation [79, 43].

▶ **Lemma 11** (Rutten [79]). *Let (V_1, γ_1) and (V_2, γ_2) be a coalgebras, and $v_1 \in V_1$ and $v_2 \in V_2$ be states. If T preserves* weak pullbacks*, then $v_1 \underline{\leftrightarrow} v_2$ if and only if v_1 and v_2 are behaviourally equivalent.*

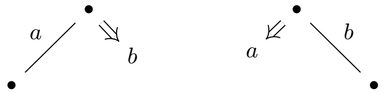
Many of the covarietors familliar to computer scientists preserve weak pullbacks, including every polynomial functor, the covariant powerset functor $\mathcal{P}$ and its κ-accessible variants $\mathcal{P}_\kappa$, and every product, coproduct, and composition of these functors [41]. Among the resulting class of functors are the deterministic automaton functor $B \times \mathrm{Id}^A$ and the nondeterministic automaton functor $B \times \mathcal{P}(\mathrm{Id})^A$ for any output set B. The added assumption that T preserve weak pullbacks leads to a rich coalgebraic theory, and much of [79] depends on it.

Under the additional assumption that T preserves weak pullbacks, Gumm and Schröder obtain the following characterisation of behavioural covarieties.

▶ **Theorem 12** (Gumm and Schröder [45]). *Let T preserve weak pullbacks. Then a structural covariety C in $\mathsf{CoAlg}(T)$ is behavioural if and only if it is* closed under total bisimulations*, i.e. if $(V_1, \gamma_1) \in \mathsf{C}$ and there is a bisimulation (R, ρ) between (V_1, γ_1) and (V_2, γ_2) such that π_1 and π_2 are surjective, then $(V_2, \gamma_2) \in \mathsf{C}$ as well.*

▶ **Example 13.** Consider the functor $T_{alt} = (\{\checkmark\} + \mathrm{Id})^{\{a,b\}}$. In a T_{alt}-coalgebra (V, γ), write $v \xrightarrow{a} v'$ if $v' = \gamma(v)(a)$ and $v \Rightarrow a$ if $\checkmark = \gamma(v)(a)$, and similarly for b. The functor T_{alt} preserves weak pullbacks, and the final T_{alt}-coalgebra is the set of all $\{a, b\}$-decorated trees t such that every node n of t has at most one $n \xrightarrow{a} n'$ transition and at most one $n \xrightarrow{b} n'$, and all other transitions are of the form $n \Rightarrow a$ or $n \Rightarrow b$. The coalgebra structure δ is the obvious parent-child transition structure.

The class of *sequence* coalgebras, consisting all those T_{alt}-coalgebras (V, γ) such that $|\gamma^{-1}(\checkmark)| = 1$, is closed under total bisimulations. By Theorem 12, the covariety of sequence coalgebras is determined by a set W_{seq} of behaviours. One way to describe this coequation is as follows: W_{seq} is the set of all behaviours t such that the first *layer* of t is one of



Of course, many of the trees in W_{seq} are *not* behaviours exhibited by sequence T_{alt}-coalgebras, as nodes from deeper layers might accept too many or too few of a, b. This can easily be fixed once we have made the following observation.

▶ **Lemma 14** (Rutten [79]). *Let $(V, \gamma), (V', \gamma')$ be T-coalgebras and $h : V \to V'$ be a T-coalgebra homomorphism. Then $h(V)$ is a subcoalgebra of (V', γ').*

Consequently, if $(V, \gamma) \models W$ for some $W \subseteq Z$, then $!_V(V)$ is a subcoalgebra of (Z, δ) contained in W. It follows from $\mathsf{CoAlg}(T)$ being cocomplete and having image factorisations that an arbitrary union of subcoalgebras of a fixed coalgebra is a subcoalgebra. In particular, there is a largest subcoalgebra $\Box W$ contained in W, and it satisfies $\mathsf{Cov}(\Box W) = \mathsf{Cov}(W)$ for any $W \subseteq Z$. In fact, $\Box W$ is the final object of $\mathsf{Cov}(\Box W)$! The operator $\Box$ is known as the "henceforth" operator in [51], and is studied in more general settings by Hughes in [48].

4.2 Beyond behaviour

A *predicate coequation in X colours* is a set of *X-patterns*, or a subset of the cofree coalgebra CX. A coalgebra (V, γ) *satisfies* the predicate coequation $W \subseteq CX$ if, for any colouring $c : V \to X$, the homomorphism $\hat{c} : (V, \gamma) \to CX$ induced by the adjunction $U \dashv C$ factors through W (that is, $\hat{c}(V) \subseteq W$).

▶ **Theorem 15** (Rutten [79], Gumm [41]). *Let T be a covarietor, X be a set, and $W \subseteq UCX$. The class $\mathsf{Cov}(W) = \{(V,\gamma) \mid (\forall c : V \to X)\ \hat{c}(V) \subseteq W\}$ is closed under subcoalgebras, coproducts, and homomorphic images.*

Under mild assumptions on T, every structural covariety is presentable by a coequation. The key to proving the converse to Theorem 15 is to give an upper bound on the number of colours that only depends on T. This is possible when T is *κ-bounded* for some cardinal κ, meaning that for any coalgebra (V,γ) and any state $v \in V$, there is a subcoalgebra S of (V,γ) such that $v \in S$ and $|S| \leqslant \kappa$. An endofunctor on Set is *bounded* if it is κ-bounded for some κ.[4] The class of bounded functors is broad enough to capture most functors in everyday use by computer scientists [79, 42]. At the beginning we assumed that T is a covarietor, but boundedness actually implies this.

▶ **Theorem 16** (Kawahara & Mori [57]). *If $T : \mathsf{Set} \to \mathsf{Set}$ is bounded, then T is a covarietor.*

Let C be a structural covariety and κ an infinite cardinal, and assume that T is κ-bounded. Given a state v of a coalgebra (V,γ), there is a subcoalgebra S of (V,γ) containing v such that $|S| \leqslant \kappa$. By a simple renaming of states, S is isomorphic to a coalgebra whose state space is a set of numbers in κ. This means that every T-coalgebra is locally isomorphic to a coalgebra of the form (S,σ) where $S \subseteq \kappa$, and in particular that every coalgebra in C is locally of this form. Writing $\mathsf{G} = \{(S,\sigma) \in \mathsf{C} \mid S \subseteq \kappa\}$, the coequation $W_{\mathsf{G}} = \bigcup\{\hat{c}(S) \mid (S,\sigma) \in \mathsf{G} \text{ and } c : S \to \kappa\}$ determines C, meaning that $\mathsf{C} = \mathsf{Cov}(W_{\mathsf{G}})$.

▶ **Theorem 17** (Rutten [79]). *If T is κ-bounded and C is a structural covariety in $\mathsf{CoAlg}(T)$, then $\mathsf{C} = \mathsf{Cov}(W)$ for some $W \subseteq C\kappa$.*

Behavioural coequations are instances of predicate coequations: They are the coequations in 1 colour. We have already seen in Examples 8, 10, and 13 that some covarieties are presentable by behavioural coequations despite the type functor failing to be 1-bounded.[5] The bound on the number of colours provided by Theorem 17 is rarely optimal in practice.

Towards a better bound, recall the definition of behavioural equivalence from §4.1. Let C be a covariety that is *closed under behavioural equivalence*, meaning that it contains every coalgebra (V,γ) such that every state $v \in V$ is behaviourally equivalent to a state of a coalgebra in C. Then C is necessarily behavioural. Indeed, if $W_{\mathsf{C}} = \{!_S(x) \mid x \in S, (S,\sigma) \in \mathsf{C}\}$, then $(S,\sigma) \models W_{\mathsf{C}}$ for any $(S,\sigma) \in \mathsf{C}$. Conversely, every state of a coalgebra satisfying W_{C} behaves like a state from a coalgebra in C, and by assumption such a coalgebra must be in C.

This argument generalizes to the λ-pattern situation by saying that a class C of coalgebras is *closed under λ-pattern equivalence* if $(V,\gamma) \in \mathsf{C}$ whenever the following condition is met: for any $v \in V$ and any colouring $c : V \to \lambda$, there is a $(S,\sigma) \in \mathsf{C}$ with a state $x \in S$ and a colouring $c' : S \to \lambda$ such that $\hat{c}(v) = \hat{c}'(x)$. We thus have:

▶ **Proposition 18.** *Let C be a covariety closed under λ-pattern equivalence. Then $\mathsf{C} = \mathsf{Cov}(W)$ for some $W \subseteq UC\lambda$.*

▶ **Example 19.** Consider the deterministic automaton endofunctor $T_{det} = 2 \times \mathrm{Id}^A$ from Example 8. Fix two words $w_1, w_2 \in A^*$ and let C be the class of deterministic automata (V,γ) in which $v \xrightarrow{w_1} v'$ and $v \xrightarrow{w_2} v''$ implies $v' = v''$. Consider an automaton $(V, \langle o, \partial\rangle)$, $v \in V$, and $c : V \to 2$ the colouring $c(x) = 1 \iff x = \partial(v)(w_1)$,[6] and assume there is an automaton

[4] Equivalently, T is *accessible* or *small* [6].

[5] It is straightforward to check that each is ω-bounded, on the other hand.

[6] Here, $\partial(v) : A^* \to V$ is defined by $\partial(v)(\epsilon) = v$ and $\partial(v)(wa) = \partial(\partial(v)(w))(a)$.

$(V', \langle o', \partial' \rangle) \in \mathsf{C}$ with $v' \in V'$ and a colouring $c' : V' \to 2$ such that $\hat{c}(v) = \hat{c}'(v')$. Then $c(\partial(v)(w_2)) = c'(\partial'(v')(w_2)) = c'(\partial'(v')(w_1)) = c(\partial(v)(w_1))$, so that $\partial(v)(w_2) = \partial(v)(w_1)$ by construction of c. It follows that C is closed under 2-pattern equivalence, and is therefore determined by a coequation in 2 colours. Indeed, where ε is the counit, it is given by

$$W = \{t \in UC2 \mid \varepsilon_2(\partial(t)(w_1)) = \varepsilon_2(\partial(t)(w_2))\}.$$

In §4.1, we saw that bisimilarity and behavioural equivalence coincide when T preserves weak pullbacks. This gave way to Theorem 12, which characterised behavioural covarieties in terms of total bisimilarity. Adámek applies the same reasoning in [4] to give a bound like the one in Prop. 18 in terms of bisimulations: For a cardinal λ, say that a covariety C is *closed under λ-colour bisimilarity* if $(V, \gamma) \in \mathsf{C}$ whenever the following condition is met: For any $c : V \to \lambda$ there is a $(S, \sigma) \in \mathsf{C}$, a colouring $c' : V' \to \lambda$, and a total bisimulation (R, ρ) between (V, γ) and (V', γ') such that $c \circ \pi_1 = c' \circ \pi_2$.

▶ **Theorem 20** (Adámek [4]). *Suppose T is a covarietor that preserves weak pullbacks. A covariety in $\mathsf{CoAlg}(T)$ is presentable by a predicate coequation in λ colours if and only if it is closed under λ-colour bisimilarity.*

▶ **Example 21.** The class C of *simple* graphs (of finite degree) can be seen as a covariety in $\mathsf{CoAlg}(\mathcal{P}_\omega)$. In a $\mathcal{P}_\omega$-coalgebra (V, γ), write $v_1 \to v_2$ to denote $v_2 \in \gamma(v_1)$. Then (V, γ) is a *simple graph* if $\to$ is a reflexive symmetric relation on V.[7] Reflexivity and symmetry are given by the coequations $W_{ref} = \{t \mid (\exists s)\ \varepsilon_2(s) = \varepsilon_2(t) \text{ and } t \to s\}$ and $W_{sym} = \{t \mid (\forall s)\ (t \to s \implies (\exists u)\ \varepsilon_2(u) = \varepsilon_2(t) \text{ and } s \to u)\}$ respectively. The modal logician will recognise the axioms (T) $p \to \Diamond p$ and (B) $p \to \Box\Diamond p$, the roles of the propositional variable p being played by the colouring map ε_2. The coequation we are looking for is therefore $W_{sim} = W_{ref} \cap W_{sym}$. By Theorem 20, C is closed under 2-colour bisimilarity. This covariety is also not behavioural: $\bullet \circlearrowleft$ and $\bullet \to \bullet \to \bullet \to \cdots$ are bisimilar, for example.

▶ **Example 22.** Recall that a coalgebra (V, γ) is *locally finite* if for any $v \in V$, there is a subcoalgebra S of (V, γ) such that $v \in S$ and S is finite. If T preserves weak pullbacks, then the class C_ω of locally finite T-coalgebras is a covariety in ω colours.

To see why, let (V, γ) be a coalgebra such that for any $c : V \to \omega$, there is a locally finite (V', γ'), a colouring $c' : V' \to \omega$, and a total bisimulation (R, σ) between (V, γ) and (V', γ') such that $c \circ \pi_1 = c' \circ \pi_2$. If $v \in V$ and $c : V \to \omega$ is any colouring, and we take $(V', \gamma'), (R, \rho), c : V' \to \omega$ as before, then vRv' for some $v' \in V'$. Since (V', γ') is locally finite, $c'(S')$ is finite for some subcoalgebra S' of (V', γ') containing v'. The projection π_2 is surjective, so $P = \pi_2^{-1}(S')$ is a subcoalgebra of (R, ρ) containing (v, v'). Taking images, we see that $c(\pi_1(P)) = c'(\pi_2(P)) = c'(S')$, so that $\pi_1(P)$ is a subcoalgebra of (V, γ) with finite image under c. Since T preserves weak pullbacks, there is a smallest subcoalgebra $\langle v \rangle$ of (V, γ) containing v [79]. This subcoalgebra is contained in every P as constructed above, so has finite image under c for any c. A set is finite if and only if every image of the set under a map into ω has a finite image, so $\langle v \rangle$ must be finite. It follows that (V, γ) is locally finite, so by Theorem 20, C_ω is presentable with a coequation in ω colours. The desired coequation consists of those ω-patterns in which only finitely many colours appear.

[7] A simple graph in this sense contains the same information as the more traditional concept from combinatorics. However, $\mathcal{P}_\omega$-coalgebra homomorphisms are not graph homomorphisms. For a coalgebraic depiction of traditional directed graphs, see pg. 22 of [80], or [55].

4.3 Logic and Avoiding Patterns

As we have already seen, it is possible for distinct coequations, like $\Box W$ and W in Example 13 for instance, to specify the same covariety. In this short section, we describe Adámek and Schwencke's framing of this equivalence between coequations as a logical equivalence in [4, 89, 90], and discuss Adamék's sound and complete deduction system for the resulting logic of coequations for a polynomial endofunctor T.

Given two coequations W_1 and W_2, W_1 is said to *imply* W_2, written $W_1 \models W_2$, if for any coalgebra (V, γ), $(V, \gamma) \models W_2$ whenever $(V, \gamma) \models W_1$. For example, $W_1 \subseteq W_2$ implies $W_1 \models W_2$, and $W \models \Box W$ and $\Box W \models W$. The inference relation $\models$ also interacts with recolourings: every $h : X \to X$ induces $\hat{c} : CX \to CX$ such that $W \models \hat{h}(W)$.

Further analysis of the inference relation $\models$ is possible with a notation used by Gumm. In [41], Gumm gives a negative description of coequations, as predicates of the form $\boxminus t = (CX) - \{t\}$ for a pattern t. This is a particularly useful notation for coequations when patterns are easily described but general predicates are not. Such is the case when T is a polynomial functor, as patterns are identifiable with certain trees.

Fix a polynomial functor $T_\Sigma = \bigcup_{p \in \Sigma} \mathrm{Id}^{ar(p)}$, where κ is a cardinal, Σ is a set, and $ar : \Sigma \to \kappa$. For a set of colours X, an X-pattern is a tree t in which every node n is labelled with a pair $(p, x) \in \Sigma \times X$ and a transition function that maps each $\alpha < ar(p)$ to each child of n. The structure map of CX is given by parent→child transitions: if n is a node of t and n' is the αth child of n, then $t \xrightarrow{\alpha} s$ when s is the subtree of t rooted at n'. There is a unique node of t with no incoming transitions, called its *root*. Each node of t is the root a tree, and we call trees of this form *subtrees* of t. We write $s \sqsubseteq t$ to denote that s is a subtree of t.

Given $t, s \in UCX$, if $s \sqsubseteq t$ and $(V, \gamma) \models \boxminus s$, then $(V, \gamma) \models \boxminus t$ as well. This is because, if $\hat{c}(v) = t$ for some colouring $c : V \to X$ and $v \in V$, then every path $t \xrightarrow{\alpha_1} t_1 \to \cdots \to t_{n-1} \xrightarrow{\alpha_n} s$ is witnessed in (V, γ) by a path $v \xrightarrow{\alpha_1} v_1 \to \cdots \to v_{n-1} \xrightarrow{\alpha_n} u$ such that $\hat{c}(u) = s$.[8] Furthermore, if s is a *recolouring* of t, i.e. $s = \hat{k}(t)$ for some colouring $k : UCX \to X$, then $(V, \gamma) \models \boxminus s$ implies $(V, \gamma) \models \boxminus t$ as well. This is due to the composition $c' = k \circ \hat{c}$, where $c : V \to X$ is any colouring of (V, γ), since $s = \hat{c}'(v)$ when $t = \hat{c}(v)$. We obtain the following proof rules.

$$\frac{\vdash \boxminus s \qquad t \to s}{\vdash \boxminus t}\ \textbf{child} \qquad\qquad \frac{\vdash \boxminus s \qquad s = \hat{k}(t)}{\vdash \boxminus t}\ k\textbf{-rec}$$

For any $t, s \in UCX$, if $\vdash \boxminus t$ can be deduced from $\vdash \boxminus s$ with k**-rec** (here, k is allowed to vary) and **child**, we write $\boxminus s \vdash \boxminus t$.

▶ **Theorem 23** (Adámek [4]). *For a polynomial endofunctor T_Σ, a set X, and any $s, t \in CX$, $\boxminus s \models \boxminus t$ if and only if $\boxminus s \vdash \boxminus t$.*

As shown in Adámek's [4] and Schwencke's [89, 90], the logic of coequations for polynomial functors (described above) can be extended to include many bounded functors. The extended logic relies on the fact that every bounded functor is a natural quotient of some polynomial functor [8], and by extension every cofree coalgebra for a bounded functor is a quotient of a cofree coalgebra for a polynomial functor. The subtree and recolouring rules apply to representatives, and with the right natural quotient[9] $T_\Sigma \Rightarrow T$, the ensuing logic is sound and complete with respect to coequational reasoning.

[8] Here, $v \xrightarrow{\alpha} u$ in (V, γ) if $u = \gamma(v)(\alpha)$.

[9] Namely, a so-called *regular presentation*. See [90] for details.

4.4 Generalized coequations

If T is not bounded, T is likely not a covarietor. In such a case, we cannot always use predicates to specify classes of coalgebras over the base category Set. However, as Aczel and Mendler showed in [2], every endofunctor on Set extends to a covarietor on the category of classes. By *approximating* cofree coalgebras, which may be proper classes in the case that T is unbounded, Adámek recovers *generalized coequations* in [3], and shows they are sufficient for specifying structural covarieties in general.

To approximate the cofree coalgebra in X colours, we follow Barr in [16] and construct its *final sequence*, the ordinal-indexed diagram

$$\begin{tikzcd} X_0 & X_1 \arrow[l, "\phi_0^1"'] & X_2 \arrow[l, "\phi_1^2"'] & \cdots \arrow[l] & X_\omega \arrow[l] & X_{\omega+1} \arrow[l, "\phi_\omega^{\omega+1}"'] & \cdots \arrow[l] \end{tikzcd}$$

Here, $X_0 = 1$ and $\phi_0^1 = !$, $X_{\alpha+1} = X \times TX_\alpha$, and $\phi_{\alpha+1}^{\beta+1} = \mathrm{id}_X \times T(\phi_\alpha^\beta)$ for any ordinals $\alpha < \beta$, and $(X_\lambda, \{\phi_\alpha^\lambda\}_{\alpha<\lambda}) = \varprojlim\{\phi_\alpha^\beta : X_\beta \to X_\alpha \mid \alpha < \beta < \lambda\}$ for λ a limit ordinal.

For any T-coalgebra (V, γ) and any colouring $c : V \to X$, let $c_0 = ! : V \to X_0$ be the unique such function, and define $c_{\alpha+1} = \langle c, T(c_\alpha) \circ \gamma\rangle$ at successor ordinals, and $c_\lambda : V \to X_\lambda$ to be the unique cone homomorphism $\{c_\alpha\}_{\alpha<\lambda} \to \{\phi_\alpha^\lambda\}_{\alpha<\lambda}$ when λ is a limit ordinal. A *generalized X-pattern* is an ordinal indexed sequence $\{t_\alpha\}_{\alpha\in\mathsf{Ord}}$ such that $t_\alpha \in X_\alpha$, and $t_\alpha = \phi_\alpha^\beta(t_\beta)$ for any $\alpha < \beta$. A *generalized coequation* is a class W of generalized patterns, and $(V, \gamma) \models W$ if for any $c : V \to X$ and $v \in V$, we find $\{c_\alpha(v)\}_{\alpha\in\mathsf{Ord}} \in W$.

▶ **Theorem 24** (Adámek [3])**.** *For any endofunctor T, a class C of T-coalgebras is a structural covariety if and only if there is a generalized coequation W such that $\mathsf{C} = \mathsf{Cov}(W)$.*

Generalized coequations are indeed generalisations of coequations for a covarietor. If T is a covarietor, then $\phi_\lambda^{\lambda+1}$ is an isomorphism for some ordinal λ [7].[10] As $\phi_\lambda^{\lambda+1}$ is an isomorphism, it has an inverse $\langle k, \delta^X\rangle : X_\lambda \to X \times TX_\lambda$, and the cofree coalgebra CX is precisely (X_λ, δ^X). In this setting, c_λ and $\hat{c}$ coincide, and the satisfaction relation from §4.2 coincides with the satisfaction of generalized coequations. Note, however, that while the set of colours is fixed in Theorem 17, the colours appearing in Theorem 24 can vary.

▶ **Example 25.** The powerset functor $\mathcal{P}$ is not bounded, as no $\phi_\lambda^{\lambda+1}$ can be a bijection. Nevertheless, the class of simple graphs from Example 21 forms a covariety. The presenting coequation is 2-coloured and can be visualised as a subset of X_3. Here, $X_3 = 2\times\mathcal{P}(2\times\mathcal{P}(2\times 2))$, so elements of X_3 can be thought of as extensional trees with 2-coloured nodes and height at most 3. The desired subset, call it W, is obtained by restricting the coequation in Example 21 to such trees. The generalized coequation W_{sim} then consists of all ordinal-indexed sequences $\{t_\alpha\}_{\alpha\in\mathsf{Ord}}$ such that $t_3 \in W$.

▶ **Example 26.** For a set V, let $\mathcal{F}V$ be the set of *filters* on V, upwards-closed subsets of $\mathcal{P}(V) - \{\varnothing\}$ that are closed under pairwise intersection. For a function $f : V \to V'$, let $\mathcal{F}(f)(F) = [f(F)]_{\mathrm{fil}}$ be the smallest filter containing $\{f(s) \mid s \in F\}$. Then $\mathcal{F}$ is an unbounded functor. As Gumm points out in [42], the category Top of topological spaces and open continuous maps is a covariety of $\mathcal{F}$-coalgebras. The structure map of a topological space sends every point to its filter of neighbourhoods. The coequation presenting Top appears in [68] in modal form, but in principle can be translated into a generalized coequation.

[10] Worrell shows in [96] that if T is κ^+-bounded, $\phi_\lambda^{\lambda+1}$ is an isomorphism when $\lambda = \kappa + \kappa$.

5 Coequations-as-equations

Our main sources for this section are [50, 47, 20, 78]. The last two papers are written in the language of *visible and hidden sorts*, making them relatively difficult to read. Here, we follow the single-sorted setup of [50, 47]. The generalisation to multiple sorts (i.e. to the category Set^S for some set of sorts S) presents only notational difficulties.

Following [50], let $\mathtt{At}$ be a set of atomic types and consider the grammars of types:

$$\mathtt{S} ::= \mathtt{A} \in \mathtt{At} \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{S}+\mathtt{S} \mid \mathtt{S}\times\mathtt{S} \qquad\qquad \mathtt{T} ::= \mathtt{A} \in \mathtt{At} \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{T}+\mathtt{T} \mid \mathtt{T}\times\mathtt{T} \mid \mathtt{X}$$

A *destructor signature* is a set of pairs of types $\sigma_i \triangleq (\mathtt{S}_i, \mathtt{T}_i), i \in I$, called *destructors*. The interpretation of a type is determined inductively given interpretations $[\![\mathtt{A}]\!]$ of $\mathtt{A} \in \mathtt{At}$ and $[\![\mathtt{X}]\!]$ as sets, and by taking $+$ to be the coproduct, $\times$ the product, 0 the empty set, and 1 the set $\{0\}$ in Set. An interpretation of a destructor $\sigma \triangleq (\mathtt{S}, \mathtt{T})$ is a map $[\![\sigma]\!] : [\![\mathtt{S}]\!] \times [\![\mathtt{X}]\!] \to [\![\mathtt{T}]\!]$, and an interpretation of a destructor signature is an interpretation of each of its destructors.

As Set is Cartesian closed, every destructor can equally be interpreted as a map $[\![\sigma]\!] : [\![\mathtt{X}]\!] \to [\![\mathtt{T}]\!]^{[\![\mathtt{S}]\!]}$. This means that the interpretation of a destructor signature can be described as a coalgebra for the functor

$$T\,[\![\mathtt{X}]\!] \triangleq \prod_{i\in I} [\![\mathtt{T}_i]\!]^{[\![\mathtt{S}_i]\!]} \qquad\qquad (\mathtt{T}\text{ typically depends on }\mathtt{X}).$$

▶ **Example 27.** Jacobs provides the example of a simple class for a bank account where $\mathtt{At} = \{\mathtt{N}\}$, and the destructor signature is $\{(\mathtt{1}, \mathtt{N}), (\mathtt{N}, \mathtt{X})\}$. An interpretation of this destructor signature can be defined by choosing $[\![\mathtt{N}]\!] = \mathbb{N}$ and two maps $\mathsf{bal} : [\![\mathtt{X}]\!] \to \mathbb{N}$ and $\mathsf{credit} : \mathbb{N} \times [\![\mathtt{X}]\!] \to [\![\mathtt{X}]\!]$ returning the balance on the account and crediting the account by a given amount respectively. Alternatively, the interpretation of this destructor signature can be a coalgebra for $\mathbb{N} \times \mathrm{Id}^{\mathbb{N}}$.

The definition of a *term* for a destructor signature is fairly elastic (see *op.cit.*) but includes at least the following rules. First, define for each atomic type $\mathtt{A} \in \mathtt{At}$ a set $\mathrm{Var}_{\mathtt{A}}$ of variables of type $\mathtt{A}$. We also define a *unique* variable x of type $\mathtt{X}$. The following rules [50, 20] are used to build terms in context:

1. Variables are terms of the corresponding type: if $a \in \mathrm{Var}_{\mathtt{A}}$ then $\vdash a : \mathtt{A}, \vdash x : \mathtt{X}$.
2. If $\sigma = (\mathtt{S}, \mathtt{T})$ is in the destructor signature, then $s : \mathtt{S}, t : \mathtt{X} \vdash \sigma(s, t) : \mathtt{T}$.

Any constructions which might be useful, such as projections and coprojections or built-in functions, can be added to the grammar of terms. In the case of Example 27 it is useful to add the function $(-) + (-) : \mathtt{N} \times \mathtt{N} \to \mathtt{N}$ which allows the term $n : \mathtt{N}\, x : \mathtt{X} \vdash \mathsf{bal}(x) + n : \mathtt{N}$ to be constructed. A *coequation-as-equation* is defined as an equation $a_1 : \mathtt{A}_1, \ldots, a_n : \mathtt{A}_n, x : \mathtt{X} \vdash s = t : \mathtt{T}$ between two terms of the same type, in the same context. The *interpretation of terms* follows in the obvious way from the interpretation of the destructor signature (and any other build-in operations like $(-) + (-)$) and function composition.

As in the case of *coequations-as-corelations*, the purpose of these equations is *not* to identify terms via a quotient, but to *select* certain behaviours. The connection with corelations can be made explicit by observing that every term will be typed like $a_1 : \mathtt{A}_1, \ldots, a_n : \mathtt{A}_n, x : \mathtt{X} \vdash t : \mathtt{T}$ with a context containing a *unique* variable $x : \mathtt{X}$, and variables of atomic type. This means that its interpretation $[\![t]\!]$ can always be Curried, and since an interpretation of the destructor signature is a coalgebra $\gamma : X \to TX$, we can view a *coequation-as-equation* $s = t$ as a corelation:

$$\begin{tikzcd} X \arrow[r, shift left, "{[\![s]\!]}"] \arrow[r, shift right, "{[\![t]\!]}"'] & {[\![\mathtt{T}]\!]^{[\![\mathtt{A}_1]\!]\times\ldots[\![\mathtt{A}_n]\!]}} \end{tikzcd} \quad \text{or a pre-cocongruence} \quad \begin{tikzcd} (X,\gamma) \arrow[r, shift left, "{[\![s]\!]}"] \arrow[r, shift right, "{[\![t]\!]}"'] & C_T\,{[\![\mathtt{T}]\!]^{[\![\mathtt{A}_1]\!]\times\ldots[\![\mathtt{A}_n]\!]}} \end{tikzcd}$$

Returning to Example 27, Jacobs gives the financially sound equation $n : \mathtt{N}, x : \mathtt{X} \vdash \mathrm{bal}(\mathrm{credit}(n, x)) = n + \mathrm{bal}(x)$. By interpreting the basic destructors as a coalgebra $\gamma : X \mapsto \mathbb{N} \times X^{\mathbb{N}}$, and Currying the interpretation of the two terms in the equation, we get a corelation $X \rightrightarrows \mathbb{N}^{\mathbb{N}}$ classifying behaviours according to what the functions $\lambda n. [\![\mathrm{bal}(\mathrm{credit}(n, x))]\!]$ and $\lambda n.n + [\![\mathrm{bal}(x)]\!]$ do at state x. The coequation-as-equation *selects* the bank accounts whose behaviours cannot be distinguished by these two functions.

6 Coequations-as-modal-formulas

Modal logic, and its generalisation coalgebraic modal logic, is an intuitive and powerful syntax to write *predicate coequations*. We follow the abstract formalism of [60, 61, 53], as it naturally lends itself to interpreting modal formulas as coequations. In this formalism, the syntax of a modal logic is given by an endofunctor L on some base category $\mathcal{C}$, typically either the category BA of Boolean Algebras for boolean modal logics (see *op.cit.* and [74, 93, 21]), or the category DL of Distributive Lattices (see [12, 28, 27]) for positive modal logics. The base category $\mathcal{C}$ encodes all the basic logical connectives whilst the functor L constructs terms with modal operators. Since one is typically interested in finitary logics, we also make the natural assumption that L is finitary, and in particular a varietor [7, Thm. 3.17]. We thus have a free-forgetful adjunction $F_L \dashv U_L, F_L : \mathcal{C} \to \mathsf{Alg}_{\mathcal{C}}(L)$. We also assume that there exists a free-forgetful adjunction $\mathsf{F} \dashv \mathsf{U}, \mathsf{F} : \mathcal{C} \to \mathsf{Set}$, which we write in sans-serif font to keep it distinct from the other adjunctions.

▶ **Example 28.** Normal modal logic is defined by the functor $L : \mathsf{BA} \to \mathsf{BA}$ which sends a boolean algebra A to the boolean algebra of formal terms $LA = \mathsf{F}\{\Box a \mid a \in \mathsf{U}A\}/\equiv$, where $\equiv$ is the stable congruence generated by the equations $\Box\top = \top, \Box(a \wedge b) = \Box a \wedge \Box b$. Usually, A is also freely generated, specifically $A = \mathsf{FAt}$ where At is the set of propositional variables.

Coalgebraic modal logics are interpreted in coalgebras, so let us fix an endofunctor $T : \mathcal{D} \to \mathcal{D}$ describing both the kind of carrier ($\mathcal{D}$-objects) and the kind of transition systems in which modal formulas are to be interpreted. We now need two pieces of categorical data.

1. A dual adjunction $G \dashv P, G : \mathcal{C} \to \mathcal{D}^{\mathrm{op}}$ connecting the "logical category" $\mathcal{C}$ to the "model category" $\mathcal{D}$ whose objects carry the models of the interpretation.
2. A *semantic natural transformation* $\delta : LP \to PT$ to recursively compute the semantics.

The framework of coalgebraic modal logic can thus be summarized as the categorical data:

$$\begin{tikzcd}
\mathsf{Set} \arrow[r, bend left, "\mathsf{F}"] \arrow[r, phantom, "\perp"] & \mathcal{C} \arrow[l, bend left, "\mathsf{U}"] \arrow[loop above, "L"] \arrow[r, bend left, "G"] \arrow[r, phantom, "\perp"] & \mathcal{D}^{\mathrm{op}} \arrow[l, bend left, "P"] \arrow[loop above, "T^{\mathrm{op}}"] & \delta : LP \to PT
\end{tikzcd}$$

By using P and δ, one can turn any T-coalgebra $\gamma : X \to TX$ into an L-algebra $P\gamma \circ \delta_X : LPX \to PTX \to PX$. We denote this construction, which is functorial since δ is natural, by $\widehat{P}(X, \gamma)$. The semantics can now be defined as follows: given a T-coalgebra (X, γ) and a set At of variables, a *valuation* is a map $v : \mathsf{At} \to \mathsf{U}PX$, which lifts to a $\mathcal{C}$-morphism $\hat{v} : \mathsf{FAt} \to PX$. Since PX is the carrier of $\widehat{P}(X, \gamma)$, we can re-type $\hat{v}$ as a $\mathcal{C}$-morphism $\hat{v} : \mathsf{FAt} \to U_L\widehat{P}(X, \gamma)$. Since L is a varietor, this map freely extends to a unique L-algebra morphism $[\![-]\!]_v : F_L\mathsf{FAt} \to \widehat{P}(X, \gamma)$, which recursively computes the interpretation of a modal formula in $F_L\mathsf{FAt}$ as an element of PX via the semantic transformation δ. A formula $\varphi \in F_L\mathsf{FAt}$ is said to be *satisfied at* $x \in X$ *for the valuation* v, written $(X, \gamma, x) \models_v \phi$, if $x \in [\![\phi]\!]_v$. A formula $\varphi \in F_L\mathsf{FAt}$ is said to be *valid in* (X, γ), written $(X, \gamma) \models \varphi$, if it is satisfied at every $x \in X$ and for every valuation $v : \mathsf{At} \to \mathsf{U}PX$.

▶ **Example 29.** In the case of the normal modal logic described in Example 28 we take $\mathcal{D} = \mathsf{Set}$, the dual adjunction is given by the powerset functor $\mathcal{P} : \mathsf{Set}^{\mathrm{op}} \to \mathsf{BA}$ and the ultrafilter functor $\mathcal{U} : \mathsf{BA} \to \mathsf{Set}^{\mathrm{op}}$, and models are coalgebras for the powerset functor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$. The semantic transformation $\delta : L\mathcal{P} \to \mathcal{P}\mathcal{P}$ thus turns a modal formula over a predicate on the carrier into the set of successors which must satisfy this predicate, from the perspective of the modality. It is defined by $\delta(\Box W) = \{W' \mid W' \subseteq W\}$. Given a coalgebra $\gamma : X \to \mathcal{P}X$ and a valuation $v : \mathsf{At} \to \mathsf{U}\mathcal{P}X$, it follows from the definition of δ and $[\![-]\!]_v$ that for any $\varphi \in F_L\mathsf{FAt}$, $[\![\Box\varphi]\!]_v$ is computed recursively via $[\![\Box\varphi]\!]_v = \{x \in X : \gamma(x) \subseteq [\![\varphi]\!]_v\}$, where $[\![p]\!]_v \triangleq v(p)$. This is the usual semantics for normal modal logic, rephrased coalgebraically.

A *set* of modal axioms $\Phi \subseteq \mathsf{U}U_LF_L\mathsf{FAt}$ defines a set of equations $e_1, e_2 : \Phi \rightrightarrows \mathsf{U}U_LF_L\mathsf{FAt}$, in the sense of §3.1, since each axiom $\varphi \in \Phi$ is shorthand for the equation $\varphi = \top$, i.e. $e_1(\varphi) = \varphi, e_2(\varphi) = \top$. We can then consider the variety defined by the coequalizer $q : F_L\mathsf{FAt} \twoheadrightarrow Q$ of the free extensions $\hat{e}_1, \hat{e}_2 : F_L\mathsf{F}\Phi \rightrightarrows F_L\mathsf{FAt}$, exactly as in §3.1. We obtain, immediately from the definitions, that every set of modal axioms defines a *variety*, and these axioms are valid in a coalgebra (X, γ) precisely when $\widehat{P}(X, \gamma)$ belongs to the variety.

▶ **Proposition 30.** *Using the notation above,* $(X, \gamma) \models \Phi$ *iff* $q \perp \widehat{P}(X, \gamma)$.

A set of modal axioms Φ can also be seen as a coequation-as-predicate which defines a covariety. To see this, assume that T is a covarietor, i.e. that there exists forgetful-cofree adjunction $U_T \dashv C_T, U_T : \mathsf{CoAlg}_{\mathcal{D}}(T) \to \mathcal{D}$, and consider the cofree coalgebra $C_TG\mathsf{FAt}$ over the $\mathcal{D}$-object of colours $G\mathsf{FAt}$. The reason for choosing these colours is that by using the counit ε of the adjunction $U_T \dashv C_T$, we can construct a canonical interpretation via the adjunctions $G \dashv P$ and $F_L \dashv U_L$, and the fact that $U_L\widehat{P} \simeq PU_T$:

$$U_TC_TG\mathsf{FAt} \xrightarrow{\varepsilon} G\mathsf{FAt} \quad \Leftrightarrow \quad \mathsf{FAt} \longrightarrow PU_TC_TG\mathsf{FAt} \quad \Leftrightarrow \quad F_L\mathsf{FAt} \xrightarrow{[\![-]\!]_\varepsilon} \widehat{P}C_TG\mathsf{FAt}.$$

With this canonical interpretation map we can view Φ as a coequation-as-predicate in $G\mathsf{FAt}$ colours selecting the elements $t \in C_TG\mathsf{FAt}$ such that $(C_TG\mathsf{FAt}, t) \models_\varepsilon \varphi$ for all $\varphi \in \Phi$.

▶ **Example 31.** Returning to the classical modal logic of Examples 28 and 29, we modify the semantics slightly by considering coalgebras for an accessible version of $\mathcal{P}$, e.g. we take $T \triangleq \mathcal{P}_\kappa$ with $\kappa = |\mathsf{At}|$ so that $\mathcal{P}_\kappa\mathsf{At} = \mathcal{P}\mathsf{At}$. The coalgebraic semantics is now given in terms of a covarietor. The cofree coalgebra $C_T\mathcal{U}\mathsf{FAt} \simeq C_T\mathcal{P}\mathsf{At}$ is then the set of all κ-branching strongly-extensional trees labelled by sets of propositional variables [96]. By definition of $[\![-]\!]_\varepsilon$, if $p \in \mathsf{At}$ then $(C_T\mathcal{P}\mathsf{At}, t) \models_\varepsilon p$ iff p belongs to the set of propositional variables labelling t. The semantics of modal formulas works in the expected way, namely $\Box\varphi$ holds at a tree t if φ holds at all its children (should it have any). Thus, every modal formula $\varphi \in F_L\mathsf{FAt}$ defines the subset of $[\![\varphi]\!]_\varepsilon \subseteq U_TC_T\mathcal{P}\mathsf{At}$, i.e. a *predicate coequation*. More generally, every set Φ of modal axioms defines the predicate coequation $\bigcap_{\varphi\in\Phi} [\![\varphi]\!]_{\varepsilon^T} \subseteq U_TC_T\mathcal{P}\mathsf{At}$, containing only those trees for which all formulas in Φ are satisfied.

Now, which covarieties can be defined from a coequation-as-modal-formula in the way we just sketched? An answer to this question has long been part of the canon of modal logic, and is known as the Goldblatt-Thomason theorem [18, Thm. 3.19]. A coalgebraic version of this theorem was developed by Kurz and Rosický in [69]. This theorem can only be stated for (and is therefore only applicable to) coalgebraic modal logics such that the semantic transformation $\delta : LP \to PT$ has an inverse natural transformation $\delta^{-1} : PT \to LP$[11,12].

[11] The naturality of the inverse in not strictly necessary, but makes the presentation easier, see [69].

[12] The existence of a natural transformation $\delta^{-1} : GL \to TG$ is also key to the duality between varieties and covarieties and between equations and coequations developed in [83]. It is also crucial to strong

From this inverse we can construct a natural transformation $h : GL \to TG$, called its *mate* [69]. For our purposes, it is enough to say that h and G allow us to turn every L-algebra $\alpha : LA \to A$ into a T-coalgebra $h_A \circ G\alpha : GA \to GLA \to TGA$. We denote this operation $\widehat{G}(A, \alpha)$. In some sense, it is dual to the functor $\widehat{P}$ defined earlier. Given a T-coalgebra (X, γ), its *ultrafilter extension* is the T-coalgebra $\widehat{G}\widehat{P}(X, \gamma)$. A class C of T-coalgebras is *closed under ultrafilter extensions* if $(X, \gamma) \in \mathsf{C}$ implies $\widehat{G}\widehat{P}(X, \gamma) \in \mathsf{C}$. A class C of T-coalgebras *reflects ultrafilter extensions* if $\widehat{G}\widehat{P}(X, \gamma) \in \mathsf{C}$ implies $(X, \gamma) \in \mathsf{C}$.

▶ **Theorem 32** (Coalgebraic Goldblatt-Thomason Theorem [69]). *Let $T : \mathsf{Set} \to \mathsf{Set}$ preserve finite sets, and assume the existence of a natural inverse $\delta^{-1} : PT \to LP$ to the semantic transformation, then a class of T-coalgebras closed under ultrafilter extensions is definable by coequations-as-modal-formulas iff it is closed under homomorphic images, subcoalgebras, coproducts, and if it reflects ultrafilter extensions.*

7 Conclusion

In this review we have presented four types of syntaxes for "writing a coequation": *coequations-as-corelations*, which come with the special syntax of *coequations-as-equations* for functors of the type $\prod_{i \in I} A_i \times \mathsf{P}_i^{B_i}$ where each P_i is polynomial, and *coequations-as-predicates*, which come with the special syntax of *coalgebraic modal logic*. It is worth emphasising that the corelation and the predicate perspective are semantically equivalent and one can move from one to the other by taking an equalizer or a cokernel pair respectively. However, both the *syntax* and the *intuition* are different, and these aspects matter a great deal in practice.

A rule of thumb for which syntax to use in which situation might be the following. Thinking of the elements of a cofree coalgebra as generalized trees, if the aim is to specify a behaviour defined by a relationship between a tree and (some of) its children, then the corelation perspective is probably the most useful. This perspective was illustrated in §5, but can also be found in the beautiful work on stream differential equations of Hansen, Kupke and Rutten [46]. On the other hand, if the aim is to enforce or avoid a particular *behavioural pattern*, then the predicate perspective is probably the most useful.

Although much of our discussion focused on literature written decades ago, coequations continue to find new uses. It was recently observed that coequations appear in formal language theory as *varieties of languages* [13, 82], which play a dual role to monoid equations. A vastly wider perspective on this relationship was explored in subsequent work [81, 10]. For another example, a behavioural coequation appeared in a proof of the completeness of an axiomatisation of *guarded Kleene algebra with tests* (GKAT) [88], an algebraic framework for reasoning about simple imperative programs. There, the coequation is the set of behaviours specified by terms in the expression language of GKAT, much like the coequation in Example 10, and is used to present the covariety of automata that implement GKAT programs. This usage of coequations may also be possible in the context of an open problem posed by Milner [72], as a covariety implicitly appears in a recent partial solution [40].

As the examples above illustrate, the use value of coequations is emerging, slowly, from the literature. Given the scope of their applications, we hope that our synthesis of the literature will make coequations more accessible to the general computer science community.

completeness proofs in coalgebraic modal logic [25, 26].

References

1 Peter Aczel. *Non-Well-Founded Sets.* Number 14 in Lecture Notes. Center for the Study of Language and Information (CSLI), 1988.

2 Peter Aczel and Nax Mendler. A final coalgebra theorem. In *Category theory and computer science*, pages 357–365. Springer, 1989.

3 Jiří Adámek. Birkhoff's covariety theorem without limitations. *Commentationes Mathematicae Universitatis Carolinae*, 46(2):197–215, 2005.

4 Jiří Adámek. A logic of coequations. In *International Workshop on Computer Science Logic*, pages 70–86. Springer, 2005.

5 Jiří Adámek, Horst Herrlich, and George E Strecker. Abstract and concrete categories. The joy of cats, 2004.

6 Jiří Adámek and Hans-E Porst. From varieties of algebras to covarieties of coalgebras. *Electronic Notes in Theoretical Computer Science*, 44(1):27–46, 2001.

7 Jiří Adámek and Hans-E Porst. On varieties and covarieties in a category. *Mathematical Structures in Computer Science*, 13(2):201, 2003.

8 Jirí Adámek and Hans-E. Porst. On tree coalgebras and coalgebra presentations. *Theor. Comput. Sci.*, 311(1-3):257–283, 2004.

9 Jiří Adámek and Vera Trnková. *Automata and algebras in categories*, volume 37. Springer Science & Business Media, 1990.

10 Jiří Adámek, Stefan Milius, Robert S.R. Myers, and Henning Urbat. Generalized Eilenberg theorem: Varieties of languages in a category. *ACM Trans. Comput. Logic*, 20(1), 2018.

11 Steve Awodey and Jesse Hughes. The coalegebraic dual of Birkoff's variety theorem. Technical report, Carnegie Mellon University, 2000.

12 Adriana Balan, Alexander Kurz, and Jiří Velebil. Positive fragments of coalgebraic logics. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 51–65. Springer, 2013.

13 Adolfo Ballester-Bolinches, Enric Cosme-Llópez, and Jan J. M. M. Rutten. The dual equivalence of equations and coequations for automata. *Inf. Comput.*, 244:49–75, 2015.

14 Bernhard Banaschewski and Horst Herrlich. *Subcategories defined by implications.* McMaster Univ., 1975.

15 Hendrik Pieter Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics.* North-Holland, 1985.

16 Michael Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114(2):299–315, 1993.

17 Garrett Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge*, 1935.

18 Patrick Blackburn, Johan FAK van Benthem, and Frank Wolter. *Handbook of modal logic.* Elsevier, 2006.

19 Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964.

20 Corina Cîrstea. A coequational approach to specifying behaviours. *Electronic Notes in Theoretical Computer Science*, 19:142–163, 1999.

21 Corina Cîrstea, Alexander Kurz, Dirk Pattinson, Lutz Schröder, and Yde Venema. Modal logics are coalgebraic. *The Computer Journal*, 54(1):31–41, 2011.

22 Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

23 Ranald Clouston and Robert Goldblatt. Covarieties of coalgebras: comonads and coequations. In *International Colloquium on Theoretical Aspects of Computing*, pages 288–302. Springer, 2005.

24 John Horton Conway. *Regular algebra and finite machines.* Courier Corporation, 2012.

25 Fredrik Dahlqvist. *Completeness-via-canonicity for coalgebraic logics.* PhD thesis, Imperial College London, 2015 .

26 Fredrik Dahlqvist. Coalgebraic completeness-via-canonicity. In *International Workshop on Coalgebraic Methods in Computer Science*, pages 174–194. Springer, 2016.

27 Fredrik Dahlqvist and Alexander Kurz. The positivication of coalgebraic logics. In *7th Conference on Algebra and Coalgebra in Computer Science (CALCO)*, 2017.

28 Fredrik Dahlqvist and David Pym. Completeness via canonicity for distributive substructural logics: a coalgebraic perspective. In *International Conference on Relational and Algebraic Methods in Computer Science*, pages 119–135. Springer, 2015.

29 Robert Davis. Universal coalgebra and categories of transition systems. *Mathematical systems theory*, 4(1):91–95, 1970.

30 Robert Davis. Multivalued operations and universal coalgebra. *Proceedings of the American Mathematical Society*, 32(2):385–388, 1972.

31 Robert Davis. Quasi cotripleable categories,. *Proceedings of the American Mathematical Society*, 35:43–48, 1972.

32 Robert Davis. Combinatorial examples in universal coalgebra. *Proceedings of the American Mathematical Society*, 89(1):32–34, 1983.

33 Robert Davis. Combinatorial examples in universal coalgebra. iii. *Proceedings of the American Mathematical Society*, 92(3):332–334, 1984.

34 E Allen Emerson and Joseph Y Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of computer and system sciences*, 30(1):1–24, 1985.

35 W. Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2013.

36 Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for NetKAT. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015*, pages 343–355. ACM, 2015.

37 Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *International Conference on Protocol Specification, Testing and Verification*, pages 3–18. Springer, 1995.

38 Robert Goldblatt. *Logics of time and computation.* Center for the Study of Language and Information, 1987.

39 Robert Goldblatt. A comonadic account of behavioural covarieties of coalgebras. *Mathematical Structures in Computer Science*, 15(2):243–269, 2005.

40 Clemens Grabmayer and Wan Fokkink. A complete proof system for 1-free regular expressions modulo bisimilarity. *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, July 2020.

41 H Peter Gumm. Elements of the general theory of coalgebras, 1999.

42 H Peter Gumm. Functors for coalgebras. *Algebra universalis*, 45(2):135–147, 2001.

43 H Peter Gumm and Tobias Schröder. Products of coalgebras. *Algebra Universalis*, 46(1):163–185, 2001.

44 Heinz-Peter Gumm. Equational and implicational classes of coalgebras. *Theoretical Computer Science*, 260(1-2):57–69, 2001.

45 Heinz-Peter Gumm and Tobias Schröder. Covarieties and complete covarieties. *Electronic Notes in Theoretical Computer Science*, 11:42–55, 1998.

46 Helle Hvid Hansen, Clemens Kupke, and Jan Rutten. Stream differential equations: specification formats and solution methods. *arXiv preprint arXiv:1609.08367*, 2016.

47 Ulrich Hensel and Horst Reichel. Defining equations in terminal coalgebras. In *Recent Trends in Data Type Specification*, pages 307–318. Springer, 1994.

48 Jesse Hughes. Modal operators for coequations. *Electronic Notes in Theoretical Computer Science*, 44(1):205–226, 2001.

49 Jesse Hughes. *A study of categories of algebras and coalgebras.* PhD thesis, Carnegie Mellon University, 2001.

50 Bart Jacobs. Mongruences and cofree coalgebras. In *International Conference on Algebraic Methodology and Software Technology*, pages 245–260. Springer, 1995.

51 Bart Jacobs. The temporal logic of coalgebras via Galois algebras. *Mathematical Structures in Computer Science*, 12(6):875–903, 2002.

52 Bart Jacobs. *Introduction to Coalgebra*, volume 59. Cambridge University Press, 2017.

53 Bart Jacobs and Ana Sokolova. Exemplaric expressivity of modal logics. *Journal of logic and computation*, 20(5):1041–1068, 2010.

54 Peter Jipsen. Concurrent Kleene algebra with tests. In Peter Höfner, Peter Jipsen, Wolfram Kahl, and Martin Eric Müller, editors, *Relational and Algebraic Methods in Computer Science - 14th International Conference, RAMiCS 2014. Proceedings*, volume 8428 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2014.

55 Christian Jäkel. A unified categorical approach to graphs, 2015. arXiv:1507.06328.

56 Tobias Kappé, Paul Brunet, Alexandra Silva, and Fabio Zanasi. Concurrent Kleene algebra: Free model and completeness. *CoRR*, abs/1710.02787, 2017.

57 Yasuo Kawahara and Masao Mori. A small final coalgebra theorem. *Theor. Comput. Sci.*, 233(1-2):129–145, 2000.

58 Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91)*, pages 214–225. IEEE Computer Society, 1991.

59 Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In *Computer Science Logic, 10th International Workshop, CSL '96, Annual Conference of the EACSL, 1996*, volume 1258 of *LNCS*, pages 244–259. Springer, 1996.

60 Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Algebraic semantics for coalgebraic logics. *Electronic Notes in Theoretical Computer Science*, 106:219–241, 2004.

61 Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Ultrafilter extensions for coalgebras. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 263–277. Springer, 2005.

62 Clemens Kupke and Raul Andres Leal. Characterising behavioural equivalence: Three sides of one coin. In *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009. Proceedings*, volume 5728 of *LNCS*, pages 97–112. Springer, 2009.

63 Alexander Kurz. A co-variety-theorem for modal logic. *Advances in Modal Logic*, 2:367–380, 1998.

64 Alexander Kurz. Specifying coalgebras with modal logic. *Electronic Notes in Theoretical Computer Science*, 11:56–70, 1998.

65 Alexander Kurz. *Logics for coalgebras and applications to computer science.* PhD thesis, Ludwig-Maximilians-Universität München, 2000.

66 Alexander Kurz. Modal rules are co-implications. *Electronic Notes in Theoretical Computer Science*, 44(1):241–253, 2001.

67 Alexander Kurz. Specifying coalgebras with modal logic. *Theoretical Computer Science*, 260(1-2):119–138, 2001.

68 Alexander Kurz and Jirí Rosickỳ. Operations and equations for coalgebras. *Mathematical Structures in Computer Science*, 15(1):149, 2005.

69 Alexander Kurz and Jiří Rosickỳ. The Goldblatt-Thomason theorem for coalgebras. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 342–355. Springer, 2007.

70 Giulio Manzonetto and Antonino Salibra. From lambda-calculus to universal algebra and back. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008. Proceedings*, volume 5162 of *LNCS*, pages 479–490. Springer, 2008.

71 Michal Marvan. On covarieties of coalgebras. *Archivum Mathematicum*, 21(1):51–63, 1985.

72 Robin Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984.

73 Lawrence S Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1-3):277–317, 1999.

74 Dirk Pattinson. Coalgebraic modal logic: Soundness, completeness and decidability of local consequence. *Theoretical Computer Science*, 309(1-3):177–193, 2003.

75 Boris I. Plotkin and Tanya Plotkin. Universal Algebra and Computer Science. In *Fundamentals of Computation Theory, 13th International Symposium, FCT 2001. Proceedings*, volume 2138 of *LNCS*, pages 35–44. Springer, 2001.

76 Horst Reichel. An approach to object semantics based on terminal co-algebras. *Mathematical Structures in Computer Science*, 5(2):129–152, 1995.

77 Grigore Roşu. A Birkhoff-like axiomatizability result for hidden algebra and coalgebra. *Electronic Notes in Theoretical Computer Science*, 11:176–193, 1998.

78 Grigore Roşu. Equational axiomatizability for coalgebra. *Theoretical Computer Science*, 260(1-2):229–247, 2001.

79 Jan JMM Rutten. Universal coalgebra: a theory of systems. Technical report, CWI, 1996.

80 Jan JMM Rutten. Universal coalgebra: a theory of systems. *Theoretical computer science*, 249(1):3–80, 2000.

81 Julian Salamanca. Unveiling Eilenberg-type correspondences: Birkhoff's theorem for (finite) algebras + duality. *CoRR*, abs/1702.02822, 2017.

82 Julian Salamanca, Adolfo Ballester-Bolinches, Marcello M. Bonsangue, Enric Cosme-Llópez, and Jan J. M. M. Rutten. Regular varieties of automata and coequations. In *Mathematics of Program Construction - 12th International Conference, MPC 2015*, volume 9129 of *LNCS*, pages 224–237. Springer, 2015.

83 Julian Salamanca, Marcello Bonsangue, and Jurriaan Rot. Duality of equations and coequations via contravariant adjunctions. In Ichiro Hasuo, editor, *Coalgebraic Methods in Computer Science*, pages 73–93, Cham, 2016. Springer International Publishing.

84 Antonino Salibra. On the algebraic models of lambda calculus. *Theor. Comput. Sci.*, 249(1):197–240, 2000.

85 Antonino Salibra and Robert Goldblatt. A finite equational axiomatization of the functional algebras for the lambda calculus. *Inf. Comput.*, 148(1):71–130, 1999.

86 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966.

87 Hanamantagouda P Sankappanavar and Stanley Burris. *A course in universal algebra*, volume 78. Citeseer, 1981.

88 Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. *CoRR*, abs/2102.08286, 2021.

89 Daniel Schwencke. Coequational logic for finitary functors. *Electronic Notes in Theoretical Computer Science*, 203(5):243–262, 2008.

90 Daniel Schwencke. Coequational logic for accessible functors. *Information and Computation*, 208(12):1469–1489, 2010.

91 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time. *CoRR*, abs/1907.05920, 2019.

92 Moshe Y Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society, 1986.

93 Yde Venema. Algebras and coalgebras. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 331–426. North-Holland, 2007.

94 Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1992.

95 Uwe Wolter. On corelations, cokernels, and coequations. *Electronic Notes in Theoretical Computer Science*, 33:317–336, 2000.

96 James Worrell. Terminal sequences for accessible endofunctors. In Bart Jacobs and Jan J. M. M. Rutten, editors, *Coalgebraic Methods in Computer Science, CMCS 1999*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 24–38. Elsevier, 1999.