Mongruences and Cofree Coalgebras ⋆

Bart Jacobs

CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands.
Email: bjacobs@cwi.nl

Abstract. A coalgebra is introduced here as a model of a certain signature consisting of a type X with various "destructor" function symbols, satisfying certain equations. These destructor function symbols are like methods and attributes in object-oriented programming: they provide access to the type (or state) X. We show that the category of such coalgebras and structure preserving functions is comonadic over sets. Therefore we introduce the notion of a 'mongruence' (predicate) on a coalgebra. It plays the dual role of a congrence (relation) on an algebra.

An algebra is a set together with a number of operations on this set which tell how to form (derived) elements in this set, possibly satisfying some equations. A typical example is a monoid, given by a set M with operations $1 \rightarrow M$, $M \times M \rightarrow M$. Here $1 = \{\emptyset\}$ is a singleton set. In mathematics one usually considers only single-typed algebras, but in computer science one more naturally uses many-typed algebras like $1 \rightarrow \mathsf{list}(A)$, $A \times \mathsf{list}(A) \rightarrow \mathsf{list}(A)$.

Here we are not primarily interested in algebras, but in coalgebras. These consist of a set together with operations "going out" of the set; they tell how to "deconstruct" elements of the set. These kind of structures naturally arise within object-oriented programming—as explained in [16, 10], but see also [13], or [5] (where the "destructors" are called "observers"). A possible (set-theoretic) semantics for objects is to see them as pairs $\langle x \in X, c: X \rightarrow T(X)\rangle$ where c is a coalgebra of a functor $T: \mathbf{Sets} \rightarrow \mathbf{Sets}$ interpreting the methods, and x is an element of the carrier X of the coalgebra c. Usually c will be a tuple $c = \langle c_1, \ldots, c_n\rangle$ of maps. Message passing then means application of the appropriate component of c to $x \in X$.

This picture is very elementary but clear enough to serve as our motivation. Here we concentrate on equations in such coalgebras. The main conceptual novelty that we introduce is the notion of what we call a *mongruence* (predicate). A mongruence plays the same role in coalgebra that a congruence plays in algebra: it is a predicate which is closed under the coalgebra operations, just like a congruence is a relation which is closed under the algebra operations.

We use these mongruences in particular to construct cofree coalgebras satisfying certain equations. In Mac Lane's book on category theory [14] (but see also [15]) there is a section VI, 8 called "Algebras are T-algebras". It is shown

⋆ To appear in Proceedings of AMAST 1995, to be published as a Springer Lecture Notes in Computer Science.

there that algebras as sets with operations satisfying certain equations are monadic over sets. The free algebra is obtained by first forming the free term algebra built on the constructors, and then taking the quotient by the least congruence relation induced by the equations. Our main result Theorem 12 is the dual of this classical result: it shows that equationally defined coalgebras are comonadic over sets. The cofree construction also proceeds in two stages: first the cofree coalgebra on the destruvctors is formed, and then one takes the sub-coalgebra given by the greatest mongruence induced by the equations.

What we present here is a fundamental study. It may have ramifications into logic, algebra, category theory, datatype theory, automata theory, semantics of object-oriented programming, database theory and linguistics. A coalgebra is a fundamental notion since it gives us a formalization of a state as a "black box" to which we only have access via specified operations. No information is given as to what is in the black box.

An explanation of the notions of mongruence and congruence (and also of induction and coinduction) at the proper level of generality requires categorical logic (in terms of fibred categories). Since we only deal with interpretation in sets here, and since we don't wish to put off too many readers, we shall be describing these notions more concretely in the logic of sets. But the notation we use is a categorical one. This 'compromise' has the advantage that readers familiar with fibred category theory can see the underlying abstract level, and readers without this familiarity can compute in sets.

1 Constructor and destructor signatures

Let At be some non-empty set, the elements of which will be regarded as atomic types. Let X be an element, not in At. We write $\overline{\mathrm{At} \cup \{X\}}$ for the formal closure of the set $\mathrm{At} \cup \{X\}$ under finite products $(1, \times)$ and coproducts $(0, +)$. The elements of this set $\overline{\mathrm{At} \cup \{X\}}$ will be regarded as types. A **constructor signature** is then essentially just a type, say $\tau \in \overline{\mathrm{At} \cup \{X\}}$, sometimes written as $\tau(X)$ to express that X may occur in τ. We think of τ as the input type of a constructor $\mathsf{constr}\colon \tau(X) \to X$. We usually assume that τ is in "disjunctive normal form", i.e. of the form $\tau \equiv \tau_1 + \cdots + \tau_n$ with τ_i consisting of a finite product $\alpha_1 \times \cdots \times \alpha_m \times X \times \cdots \times X$ of atoms α_i and X's. The single constructor $\mathsf{constr}\colon \tau \to X$ may then be decomposed into n constructors $\mathsf{constr}_i\colon \tau_i \to X$. For example the constructor signature of monoids is given by the type $\tau(X) = 1 + X \times X$ (with constructors $1 \to X$ and $X \times X \to X$ for unit and multiplication) and the signature for binary α-labelled trees by the type $\tau(X) = 1 + X \times \alpha \times X$ (with constructors $1 \to X$ for the empty tree, and $X \times \alpha \times X \to X$ for forming a node).

(Such a signature is called **single-typed** if its set At of atomic types is empty. For example, the above signature of monoids is single-typed.)

Given such a signature, one can form typed terms $v_1\colon \sigma_1, \ldots, v_n\colon \sigma_n \vdash M\colon X$ starting from the typed variables $v_i\colon \sigma_i$, the constructors from the signature and the term forming operation as associated with finite products and coproducts:

tupling and projections and cotupling and coprojections (including empty ones), see e.g. [12, 6]. We use capital letters Γ to abbreviate contexts $v_1 : \sigma_1, \ldots, v_n : \sigma_n$ of variable declarations. An **equation** is then a pair of terms with type X in the same context, written as $\Gamma \vdash M =_X M'$. Given a set $\mathcal{E}$ of such equations, we can take these as axioms, and start equational logic. There one has as derivation rules that $=_\sigma$ is reflexive, symmetric and transitive, and also that it is compatible: if $\Gamma \vdash M =_\sigma M'$ is derivable, then for each term $\Gamma, v{:}\,\sigma \vdash N{:}\,\rho$, the equation $\Gamma \vdash N[M/v] =_\rho N[M'/v]$ is also derivable. Finally, one has the substitution rule: if $\Gamma \vdash M{:}\,\sigma$ and $\Gamma, v{:}\,\sigma \vdash N =_\rho N'$ are derivable, then so is $\Gamma \vdash N[M/v] =_\rho N'[M/v]$.

A **destructor signature** has "destructors" where a constructor signature has constructors. Instead of telling how to build up X's (as above), a destructor tells us how to act on X. Destructors have the form $X \to \tau(X)$. But we shall consider them with parameter type σ in $\sigma \times X \to \tau(X)$, with X not occuring in σ. Equivalently, we can write it as $X \to (\sigma \Rightarrow \tau(X))$ using the exponent type former $\Rightarrow$. Of course, σ may be the singleton type 1, in which case it can be ignored.

Formally a destructor signature is given by a set At of atomic types, by an n-tuple $\tau_1, \ldots, \tau_n \in \overline{\mathrm{At} \cup \{X\}}$ of "output" types in which X may occur, and by an n-tuple $\sigma_1, \ldots, \sigma_n \in \overline{\mathrm{At}}$ of "input" or "parameter" types, in which X does not occur. We may name the destructors explicitly as in

$$\begin{array}{lcl} \mathsf{destr}_1 : \sigma_1 \times X \longrightarrow \tau_1(X) & & \mathsf{destr}_1 : X \longrightarrow \sigma_1 \Rightarrow \tau_1(X) \\ \vdots & \text{or as in} & \vdots \\ \mathsf{destr}_n : \sigma_n \times X \longrightarrow \tau_n(X) & & \mathsf{destr}_n : X \longrightarrow \sigma_n \Rightarrow \tau_n(X) \end{array}$$

using the adjoint correspondence between $\sigma \times X \to Y$ and $X \to (\sigma \Rightarrow Y)$. This is also called 'Currying'. The type of such a destructor signature is

$$\tau(X) = (\sigma_1 \Rightarrow \tau_1(X)) \times \cdots \times (\sigma_n \Rightarrow \tau_n(X)).$$

Examples of destructor signatures are

$$\mathsf{head}{:}\, X \longrightarrow \alpha, \qquad \mathsf{tail}{:}\, X \longrightarrow X$$

where one may think of X as lists (of elements of type α). A very rudimentary bank account (of a single person) may be described as

$$\mathsf{bal}{:}\, X \longrightarrow \mathsf{N}, \qquad \mathsf{credit}{:}\, \mathsf{N} \times X \longrightarrow X.$$

And a deterministic automaton with alphabet type Σ and output type U has two destructors

$$\mathsf{out}{:}\, X \longrightarrow U, \qquad \mathsf{act}{:}\, \Sigma \times X \longrightarrow X.$$

(The restriction that X should not occur in parameter types σ is a severe one; it precludes a specification of points with a distance operation, as in

$$\mathsf{hor\_coord}{:}\, X \longrightarrow \mathsf{R}, \qquad \mathsf{vert\_coord}{:}\, X \longrightarrow \mathsf{R}, \qquad \mathsf{dist}{:}\, X \times X \longrightarrow \mathsf{R}.$$

Having such X's in parameter types introduces contravariance problems for exponents—which we wish to avoid at this stage. See also the discussion in [10] on this matter.)

A destructor $\mathsf{destr}: \sigma \times X \to \tau(X)$ will be called an **attribute** if X does not occur in the output type τ. Otherwise, it will be called a **method**.

One can also do equational logic with such destructor signatures. One takes the destructors (of a specific signature) as atomic terms, and generate a term calculus, using the term forming operations for finite product & coproduct types and exponents. An equation is then a sequent of the form $\Gamma, x: X \vdash M =_\rho M'$ where M, M' are terms of type τ in context $\Gamma, x: X$, with the restriction that X does not occur in (the types in) Γ.

For example, for the above bank account there is an obvious equation

$$n: \mathsf{N}, x: X \vdash \mathsf{bal}(\mathsf{credit}(n, x)) =_{\mathsf{N}} n + \mathsf{bal}(x).$$

(Strictly speaking, we have to regard N not as an atomic type, but N with $+$ as an imported signature; but this is not essential in what follows.)

2 Interpreting constructors and destructors

What follows applies to an arbitrary category $\mathbb{C}$ with finite product $(1, \times)$, coproducts $(0, +)$ and exponents $\Rightarrow$. Such a category is called bicartesian closed. The projections will be written as $\pi: X \times Y \to X$, $\pi': X \times Y \to Y$ and the coprojections as $\kappa: X \to X + Y$, $\kappa': Y \to X + Y$. Tupling of $f: Z \to X$ and $g: Z \to Y$ yields $\langle f, g\rangle: Z \to X \times Y$, and cotupling of $f: X \to Z$ and $g: Y \to Z$ gives $[f, g]: X + Y \to Z$.

Definition 1. A **polynomial functor** $T: \mathbb{C} \to \mathbb{C}$ is a functor which is built up in the following fashion.

(i) The identity functor $id: \mathbb{C} \to \mathbb{C}$ is polynomial.

(ii) For any object $A \in \mathbb{C}$, the constant functor $X \mapsto A$ is polynomial.

(iii) Given polynomial functors $T, S: \mathbb{C} \to \mathbb{C}$, the following functors are also polynomial,

$$X \mapsto T(X) \times S(X) \quad \text{and} \quad X \mapsto T(X) + S(X).$$

(iv) If $A \in \mathbb{C}$ and T is a polynomial functor, then

$$X \mapsto A \Rightarrow T(X)$$

is also polynomial. (This last clause will only be needed for destructor signatures.)

Let At be a set of atomic types, and assume that for each $\alpha \in \mathrm{At}$, an interpretation $A = [\![\, \alpha \,]\!] \in \mathbb{C}$ is given. Then, associated with the type $\tau(X) = \tau_1(X) + \cdots + \tau_n(X)$ of a constructor signature, there is a polynomial functor $T: \mathbb{C} \to \mathbb{C}$, which follows the structure of τ. For example, the functor associated with $1 \to X$, $\alpha \times X \to X$ is $X \mapsto 1 + A \times X$, where $A = [\![\, \alpha \,]\!]$.

Similarly, associated with type type $\tau(X) = (\sigma_1 \Rightarrow \tau_1(X)) \times \cdots \times (\sigma_n \Rightarrow \tau_n(X))$ of a destructor signature there is a polynomial functor $T: \mathbb{C} \to \mathbb{C}$. Here one really needs clause (iv) about exponents. For instance, the functor associated with the above automaton $X \to U$, $\Sigma \times X \to X$ is $X \mapsto [\![U]\!] \times ([\![\Sigma]\!] \Rightarrow X)$.

For a functor $T: \mathbb{C} \to \mathbb{C}$, an **algebra** is a "carrier" object $I \in \mathbb{C}$ together with a map $a: T(I) \to I$. And a morphism of algebras (also called 'algebra map') from $(a: T(I) \to I)$ to $(b: T(J) \to J)$ is a morphism $u: I \to J$ in $\mathbb{C}$ between the carrier objects with $u \circ a = b \circ T(u)$. This yields a category $\mathrm{Alg}(T)$. A basic observation is that an algebra of the polynomial functor T associated with a constructor signature $\mathsf{constr}_1: \tau_1(X) \to X, \ldots, \mathsf{constr}_n: \tau_n(X) \to X$ gives an interpretation $I = [\![X]\!] \in \mathbb{C}$ of X and an interpretation

$$a = [\, [\![\mathsf{constr}_1]\!], \ldots, [\![\mathsf{constr}_n]\!] \,]: T(I) \longrightarrow I$$

of all the constructors in one n-cotuple. And algebra maps are morphisms in $\mathbb{C}$ between the interpretations of X, commuting with the interpretations of the constructors.

Dually, a **coalgebra** of a functor $T: \mathbb{C} \to \mathbb{C}$ consists of a carrier object I together with a map $c: I \to T(I)$. And a map of coalgebras from $(c: I \to T(I))$ to $(d: J \to T(J))$ is a map $u: I \to J$ in $\mathbb{C}$ with $d \circ u = T(u) \circ c$. We thus get a category $\mathrm{CoAlg}(T)$. And a coalgebra of the functor T associated with a destructor signature $\mathsf{destr}_1: X \to (\sigma_1 \Rightarrow \tau_1(X)), \ldots, \mathsf{destr}_n: X \to (\sigma_n \Rightarrow \tau_n(X))$ consists of a carrier object $I = [\![X]\!] \in \mathbb{C}$ together with a map

$$c = \langle [\![\mathsf{destr}_1]\!], \ldots, [\![\mathsf{destr}_n]\!] \rangle: I \longrightarrow T(I)$$

interpreting the destructors in a single n-tuple. Again, coalgebra maps commute with these interpretations of the destructors.

Example 1. Take $\mathbb{C} = \mathbf{Sets}$, the category of sets and functions. The empty set 0 (or $\emptyset$) is initial object and $1 = \{\emptyset\}$ is terminal object. The cartesian product $X \times Y$ of two sets X, Y is categorical product, and the disjoint union $X + Y$ is coproduct. We consider the bank account destructor signature $\mathsf{bal}: X \to \mathsf{N}$, $\mathsf{credit}: \mathsf{N} \times X \to X$. Let's take the set of natural numbers $\mathbb{N}$ as interpretation of the atomic type N (and addition on $\mathbb{N}$ as interpretation of $+$ on N). The functor involved is $X \mapsto \mathbb{N} \times (\mathbb{N} \Rightarrow X)$. We mention the following examples of coalgebras of this functor.

(a) $\emptyset \mapsto \langle 0, \lambda n.\, \emptyset \rangle : 1 \longrightarrow \mathbb{N} \times (\mathbb{N} \Rightarrow 1)$
(b) $n \mapsto \langle n, \lambda m.\, n + m \rangle : \mathbb{N} \longrightarrow \mathbb{N} \times (\mathbb{N} \Rightarrow \mathbb{N})$
(c) $\alpha \mapsto \langle \mathsf{sum}(\alpha), \lambda n.\, n \cdot \alpha \rangle : \mathbb{N}^\star \longrightarrow \mathbb{N} \times (\mathbb{N} \Rightarrow \mathbb{N}^\star)$
(d) $\varphi \mapsto \langle \varphi([\,]), \lambda n.\, \lambda \alpha.\, \varphi(n \cdot \alpha) \rangle : (\mathbb{N}^\star \Rightarrow \mathbb{N}) \longrightarrow \mathbb{N} \times (\mathbb{N} \Rightarrow (\mathbb{N}^\star \Rightarrow \mathbb{N}))$.

One sees how the various interpretations of the 'balance' and 'credit' destructors are encoded in these coalgebras.

The next step is to see what validity of equations means in such (co)algebra models. Given the interpretations of the basic terms (the constructors and destructors), one can extend the interpretation to all terms, in such a way that a term in context, $\Gamma = x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n \vdash M{:}\rho$ becomes a morphism, $[\![M]\!]{:}\, [\![\Gamma]\!] \longrightarrow [\![\rho]\!]$, where $[\![\Gamma]\!] = [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$. As usual, one has that (co)algebra maps commute with these interpretations of terms (in the obvious sense).

An equation (or axiom) $\Gamma \vdash M =_\rho M'$ is then **valid** (or **holds**) if the two resulting maps $[\![M]\!], [\![M']\!]{:}\, [\![\Gamma]\!] \rightrightarrows [\![\rho]\!]$ are equal. There is the usual soundness result: all derivable equations are also valid.

In the bank account example above, the coalgebras (b) and (c) satisfy the equation mentioned at the end of the previous section.

For a suitable collection $\mathcal{E}$ of equations, we write $\mathrm{Alg}(T, \mathcal{E}) \hookrightarrow \mathrm{Alg}(T)$ and $\mathrm{CoAlg}(T, \mathcal{E}) \hookrightarrow \mathrm{CoAlg}(T)$ for the full subcategories of algebras and coalgebras satisfying the equations in $\mathcal{E}$.

In this paper we are interested in forcing the validity of equations. For algebras this is done in the familiar way via quotients: given a set of equations $\mathcal{E} = \{\Gamma_\ell \vdash M_\ell =_X M'_\ell \mid \ell \in L\}$, one takes the image of each tuple $\langle [\![M_\ell]\!], [\![M'_\ell]\!] \rangle$ as $[\![\Gamma_\ell]\!] \twoheadrightarrow R_\ell \hookrightarrow [\![X]\!] \times [\![X]\!]$, and one forms the union $R = \bigcup_\ell R_\ell \hookrightarrow [\![X]\!] \times [\![X]\!]$ of these. Quotienting $[\![X]\!]$ by the least congruence relations containing R (that is also an equivalence relation) yields the free algebra (on the given one) which satisfies the equations in $\mathcal{E}$.

In the coalgebra case, one does not quotient by a congruence, but one takes the subcoalgebra induced by what we call a **mongruence**. To explain properly what the latter is, we need to say more about the logic of predicates on sets.

3 Predicates and relations

First we define the categories **Pred** and **Rel** of predicates and relations on sets, and mention their basic properties. Then we describe how polynomial functors $T{:}\,\mathbf{Sets} \to \mathbf{Sets}$ can be lifted to polynomial functors $\mathrm{Pred}(T){:}\,\mathbf{Pred} \to \mathbf{Pred}$ and $\mathrm{Rel}(T){:}\,\mathbf{Rel} \to \mathbf{Rel}$ by induction on the structure of T. This is as in [7, 9, 8]. Algebras and coalgebras of these lifted functors $\mathrm{Pred}(T)$ and $\mathrm{Rel}(T)$ give important (logical) information about T.

Definition 2. (i) The category **Pred** has as objects pairs (I, P) where $P \subseteq I$ is a subset of I—or a predicate on I. Morphisms $(P \subseteq I) \to (Q \subseteq J)$ are functions $u{:}\, I \to J$ for which one has an implication $\forall i \in I.\, P(i) \supset Q(u(i))$.

(ii) Similarly, the category **Rel** has pairs (I, R) where $R \subseteq I \times I$ as objects. And morphisms $(R \subseteq I \times I) \to (S \subseteq J \times J)$ are functions $u{:}\, I \to J$ satisfying $\forall i, i' \in I.\, R(i, i') \supset S(u(i), u(i'))$.

There are obvious forgetful functors $\mathbf{Pred} \to \mathbf{Sets}$ and $\mathbf{Rel} \to \mathbf{Sets}$, namely $(I, P) \mapsto I$ and $(I, R) \mapsto I$. These are (split) fibrations: for each function $u{:}\, I \to J$ there are substitution functors:

$$\begin{aligned}(Q \subseteq J) &\mapsto (\{i \mid u(i) \in Q\} \subseteq I)\\ (S \subseteq J \times J) &\mapsto (\{(i, i') \mid (u(i), u(i')) \in S\} \subseteq I \times I).\end{aligned}$$

Both these functors are denoted by u^*; they are functors indeed, namely from the "fibre" poset of predicates/relations on J to the "fibre" poset of predicates/relations on I (with inclusion as ordering). The assignment $u \mapsto u^*$ is functorial: one has $id^* = id$ and $(v \circ u)^* = u^* \circ v^*$.

There are special functors $1\colon \mathbf{Sets} \to \mathbf{Pred}$ and $\mathrm{Eq}\colon \mathbf{Sets} \to \mathbf{Rel}$, which map a set $I \in \mathbf{Sets}$ to the truth predicate $1(I) = (I \subseteq I)$ on I, and to the equality relation $\mathrm{Eq}(I) = (\{(i,i) \mid i \in I\} \subseteq I \times I)$ on I.

We mention some basics in the logic of sets.

Proposition 3. (i) *Quantification: for a function $u\colon I \to J$ the substitution functor u^* has (both for predicates and for relations) a left adjoint $\coprod_u$ and a right adjoint $\prod_u$. These are given on $P \subseteq I$ and $R \subseteq I \times I$ by*

$$\begin{aligned}
\textstyle\coprod_u(P) &= \{j \in J \mid \exists i \in I.\, u(i) = j \wedge P(i)\} \\
\textstyle\prod_u(P) &= \{j \in J \mid \forall i \in I.\, u(i) = j \supset P(i)\} \\
\textstyle\coprod_u(R) &= \{(j,j') \in J \times J \mid \exists i,i' \in I.\, u(i) = j \wedge u(i') = j' \wedge\ R(i,i')\} \\
\textstyle\prod_u(R) &= \{(j,j') \in J \times J \mid \forall i,i' \in I.\, u(i) = j \wedge u(i') = j' \supset\ R(i,i')\}.
\end{aligned}$$

(The "Beck-Chevalley" conditions holds for these $\prod$'s and $\coprod$'s.)

(ii) *Subsets (or comprehension) and quotients: the truth functor $1\colon \mathbf{Sets} \to \mathbf{Pred}$ has a right adjoint $\{-\}$; it maps a predicate $(P \subseteq I)$ to its extent $\{i \in I \mid P(i)\} \in \mathbf{Sets}$. And the equality functor $\mathrm{Eq}\colon \mathbf{Rel} \to \mathbf{Sets}$ has a left adjoint $\mathcal{Q}$; it maps a relation $(R \subseteq I \times I)$ to the quotient $I/\widehat{R} \in \mathbf{Sets}$, where $\widehat{R} \subseteq I \times I$ is the least equivalence relation containing R.*

(iii) *Logical predicates and logical relations: both $\mathbf{Pred}$ and $\mathbf{Rel}$ are bicartesian closed, and the functors $\mathbf{Pred} \to \mathbf{Sets}$, $\mathbf{Rel} \to \mathbf{Sets}$ strictly preserve this structure. These BiCCC-structures are given by the so-called "logical" formulas; e.g.*

$$\begin{aligned}
(R \subseteq I \times I) \times (S \subseteq J \times J) &= (\pi \times \pi)^*(R) \wedge (\pi' \times \pi')^*(S) \\
&= \{(\langle i,j\rangle, \langle i',j'\rangle) \mid R(i,i') \wedge S(j,j')\} \\
(P \subseteq I) + (Q \subseteq J) &= \textstyle\coprod_\kappa(P) \vee \coprod_{\kappa'}(Q) \\
&= \{z \in I + J \mid (\exists i \in I.\, z = \kappa i \wedge P(i)) \\
&\qquad \vee (\exists j \in J.\, z = \kappa' j \wedge Q(j)\}.
\end{aligned}$$

See [7, 8] for a more abstract description. □

A polynomial functor $T\colon \mathbf{Sets} \to \mathbf{Sets}$ acting on sets can be lifted to two new polynomial functors $\mathrm{Pred}(T)\colon \mathbf{Pred} \to \mathbf{Pred}$ acting on predicates and to $\mathrm{Rel}(T)\colon \mathbf{Rel} \to \mathbf{Rel}$ acting on relations, in commuting squares

$$\begin{tikzcd}
\mathbf{Pred} \arrow[r, "\mathrm{Pred}(T)"] \arrow[d] & \mathbf{Pred} \arrow[d] \\
\mathbf{Sets} \arrow[r, "T"'] & \mathbf{Sets}
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
\mathbf{Rel} \arrow[r, "\mathrm{Rel}(T)"] \arrow[d] & \mathbf{Rel} \arrow[d] \\
\mathbf{Sets} \arrow[r, "T"'] & \mathbf{Sets}
\end{tikzcd}$$

One defines these functors by induction on the structure of T. One builds $\mathrm{Pred}(T)$ and $\mathrm{Rel}(T)$ with the same structure as T, except that the constant functor $X \mapsto A$ in T—for $A \in \mathbf{Sets}$—is replaced by the constant functors

$$X \mapsto 1(A) \text{ in } \mathrm{Pred}(T) \qquad \text{and} \qquad X \mapsto \mathrm{Eq}(A) \text{ in } \mathrm{Rel}(T).$$

The following results are proved by induction on the structure of the functor T.

Lemma 4. (i) *These* $\mathrm{Pred}(T)$ *and* $\mathrm{Rel}(T)$ *are fibred functors: they commute with substitution functors* u^*.

(ii) *They are also cofibred: they commute with the* $\coprod_u$*'s. (In order to handle clause (iv) on exponents in the definition of polynomial functors, one needs the Axiom of Choice.)*

(iii) $\mathrm{Pred}(T)$ *commutes with truth, and* $\mathrm{Rel}(T)$ *with equality:* $\mathrm{Pred}(T) \circ 1 = 1 \circ T$ *and* $\mathrm{Rel}(T) \circ \mathrm{Eq} = \mathrm{Eq} \circ T$. □

4 Mongruences

In this section we turn to algebras and coalgebras of the lifted functors $\mathrm{Pred}(T)$ and $\mathrm{Rel}(T)$. The following table gives an overview.

	algebra	coalgebra
$\mathrm{Pred}(T)$	induction assumption	**mongruence**
$\mathrm{Rel}(T)$	congruence	coinduction assumption
	(on the underlying T-algebra)	(on the underlying T-coalgebra)

This table defines a mongruence (predicate) $P \subseteq I$ as a coalgebra in **Pred** of the form $P \to \mathrm{Pred}(T)(P)$. If the underlying map is $c\colon I \to T(I)$, then we say that P is a mongruence on the (T-)coalgebra c.

(From [9, 8] it follows that in the logic of sets one always has induction and coinduction: for an initial T-algebra $a\colon T(I) \to I$, the truth predicate $1(I)$ on I carries $\mathrm{Pred}(T)(1(I)) = 1(T(I)) \stackrel{1(a)}{\to} 1(I)$ as initial $\mathrm{Pred}(T)$-algebra. And for a terminal coalgebra $c\colon I \to T(I)$ the equality relation $\mathrm{Eq}(I)$ on I has $\mathrm{Eq}(I) \stackrel{\mathrm{Eq}(c)}{\to} \mathrm{Eq}(T(I)) = \mathrm{Rel}(T)(\mathrm{Eq}(I))$ as terminal $\mathrm{Rel}(T)$-coalgebra. The reader unfamiliar with this (co)algebraic description of (co)induction may wish to see that for example finite-list-induction amounts to initiality of the truth predicate on the initial algebra $\mathsf{list}(A) = A^\star$ of $X \mapsto 1 + A \times X$. And coinduction for infinite lists amounts to terminality of the equality relation on the terminal coalgebra A^ω of $X \mapsto A \times X$.)

We turn to congruences and mongruences. First we notice that a relation $R \subseteq I \times I$ is a congruence on an algebra $a\colon T(I) \to I$ if and only if $\mathrm{Rel}(T)(R) \subseteq a^*(R)$, if and only if $\coprod_a(\mathrm{Rel}(T)(R)) \subseteq R$. Similarly a predicate $P \subseteq I$ is a mongruence on a coalgebra $c\colon I \to T(I)$ if and only if $P \subseteq c^*\mathrm{Pred}(T)(P)$, if and only if $\coprod_c(P) \subseteq \mathrm{Pred}(T)(P)$.

Example 2. Consider the functor $T(X) = A \times X$ on **Sets**, and a T-coalgebra $\langle h, t\rangle: I \to A \times I$. One thinks of I as a set of lists of A's, with $h: I \to A$ as head and $t: I \to I$ as tail operation. A predicate $P \subseteq I$ is then a mongruence on $\langle h, t\rangle$ if and only if for each $i \in I$ one has $i \in P$ implies $t(i) \in P$, since $\mathrm{Pred}(T)(P) = \{\langle a, i\rangle \mid a \in A \wedge t(i) \in P\} \subseteq A \times I$. Thus a mongruence is closed under application of the tail method.

Notice that for an arbitrary predicate $P \subseteq I$ there is a greatest mongruence $\underline{P} \subseteq P$, namely the predicate $\underline{P} = \{i \in I \mid \forall n \in \mathbb{N}.\, t^{(n)}(i) \in P\}$. This will be formulated more generally in the next section.

The following are basic examples of congruences and mongruences.

Proposition 5. (i) *Equality is a congruence and truth is a mongruence.*

(ii) *Kernel relations of algebra maps are congruences, and images of coalgebra maps are mongruences.*

Proof. (i) Because $\mathrm{Rel}(T)(\mathrm{Eq}(I)) = \mathrm{Eq}(T(I))$ and $\mathrm{Pred}(T)(1(I)) = 1(T(I))$.

(ii) The kernel relation of an algebra map $u: I \to J$ from a to b is the relation $\mathrm{Ker}(u) = u^*(\mathrm{Eq}(J))$. It is a congruence since

$$\begin{aligned} a^*\mathrm{Ker}(u) \;=\; a^*u^*\mathrm{Eq}(J) &= T(u)^*b^*\mathrm{Eq}(J) \\ &\supseteq T(u)^*\mathrm{Eq}(T(J)) \\ &= T(u)^*\mathrm{Rel}(T)(\mathrm{Eq}(J)) \\ &= \mathrm{Rel}(T)(u^*\mathrm{Eq}(J)) \;=\; \mathrm{Rel}(T)(\mathrm{Ker}(u)). \end{aligned}$$

The image of a coalgebra map $u: I \to J$ from c to d is $\mathrm{Im}(u) = \coprod_u 1(I)$. It is a mongruence since

$$\begin{aligned} \textstyle\coprod_d \mathrm{Im}(u) \;=\; \coprod_d \coprod_u 1(I) &= \textstyle\coprod_{T(u)} \coprod_c 1(I) \\ &\subseteq \textstyle\coprod_{T(u)} 1(T(I)) \\ &= \textstyle\coprod_{T(u)} \mathrm{Pred}(1(I)) \\ &= \mathrm{Pred}(T)(\textstyle\coprod_u 1(I)) \;=\; \mathrm{Pred}(T)(\mathrm{Im}(u)). \end{aligned}$$ □

We have already seen that the predicate lifting of T commutes with the fibration and with truth; but also with comprehension, as we shall see next.

Lemma 6. *The canonical map* $T \circ \{-\} \to \{-\} \circ \mathrm{Pred}(T)$ *is an isomorphism.*

Proof. By induction on the structure of T, using that the comprehension functor $\{-\}: \mathbf{Pred} \to \mathbf{Sets}$ preserves finite products and coproducts. For exponents we explicitly check: for $T(-) = A \Rightarrow S(-)$ and $P \subseteq I$,

$$\begin{aligned} \{\mathrm{Pred}(T)(P)\} &= \{f \in A \Rightarrow S(I) \mid \forall a.\, f(a) \in \mathrm{Pred}(S)(P)\} \\ &= A \Rightarrow \{\mathrm{Pred}(S)(P)\} \\ &\cong A \Rightarrow S(\{P\}) \qquad \text{by induction hypothesis} \\ &= T(\{P\}). \end{aligned}$$ □

Corollary 7. *A predicate $P \subseteq I$ is a mongruence on a coalgebra $c: I \to T(I)$ if and only if the extent $\{P\} \hookrightarrow I$ carries a (necessarily unique) sub-coalgebra structure c_P in*

$$\begin{tikzcd}
\{\mathrm{Pred}(T)(P)\} \arrow[d, equal, "\wr"] \arrow[dr, tail, bend left] & \\
T(\{P\}) \arrow[r, tail, "T(\pi)"'] & T(I) \\
\{P\} \arrow[u, dashed, "c_P"'] \arrow[r, tail, "\pi"] \arrow[uu, bend left=90, "\{d\}"] & I \arrow[u, "c"']
\end{tikzcd}$$

Proof. Because there are bijective correspondences between

(a) coalgebras $d: P \to \mathrm{Pred}(T)(P)$ over $c: I \to T(I)$.

(b) functions $\{d\}: \{P\} \to \{\mathrm{Pred}(T)(P)\}$ making the outer rectangle commute.

(c) functions (or subcoalgebras) $c_P: \{P\} \to T(\{P\})$ making the inner rectangle commute. □

Remark. This result is like: quotients by congruences inherit algebra structures. It is important to stress that the essence of mongruences and congruences (and also of induction and coinduction) is "logical": these notions live in **Pred** and **Rel**—and in other fibred categories where one need not have subsets $\{-\}$ or quotients—and the induced coalgebras and algebras in **Sets** are secondary.

One may be tempted to define a mongruence on a coalgebra $c: I \to T(I)$ in a seemingly equivalent (and much simpler) way as a predicate $P \hookrightarrow I$ for which there is a map c' in

$$\begin{tikzcd}
P \arrow[r, "c'"] \arrow[d, hook, "m"'] & T(P) \arrow[d, "T(m)"] \\
I \arrow[r, "c"] & T(I)
\end{tikzcd} \qquad (*)$$

The content of the above corollary is that the definition $(*)$ is the same as the earlier definition of mongruence as a $\mathrm{Pred}(T)$-coalgebra—as long as we restrict ourselves to polynomial functors T on **Sets**.

But this "subcoalgebra" definition $(*)$ has two serious drawbacks.

(i) It makes being a mongruence a structure instead of a property. One has to know that T preserves injections (or monos in an arbitrary category) so that there can be at most one such c'.

(ii) But a more serious drawback is that this definition is not "logical" in the sense that it is on the wrong level: it is not about (abstract) predicates, but about types. To explain this criticism properly one needs some fibred category theory. There may be a "base" category that one wishes to reason about in different ways (i.e. with different logics). In fibred category theory one can do this by putting different fibrations on this "base" category. By using the liftings $\mathrm{Pred}(T)$ and $\mathrm{Rel}(T)$ one also gets these different ways of reasoning about T.

For example, one may wish to reason about a base category of metric spaces M with closed subsets $X \subseteq M$, or with morphisms $M \to [0,1]$ as predicates on M. This involves different logics, which have their own "logical liftings". The formulation $(*)$ cannot take such differences into account. It is on the wrong level: in the "base" category instead of in the "total" category of the fibration, which is where the predicates live.

The definition of bisimulation by Aczel and Mendler [2] is not on the right level for the very same reasons. (These bisimulations should be what we call coinduction assumptions, see Example 4 below.) And the same applies to the definition of congruence in [15].

5 Greatest mongruences

In algebra one always finds phrases like "let $\overline{R}$ be the least congruence containing a relation R". In coalgebra we shall be using "the greatest mongruence $\underline{P}$ contained in a predicate P". In this section we investigate how to obtain such $\underline{P}$ via fixed points.

Let $T\colon \mathbf{Sets} \to \mathbf{Sets}$ be a polynomial functor and $c\colon I \to T(I)$ a coalgebra. For a predicate $P \subseteq I$ we seek the greatest mongruence $\underline{P} \subseteq I$ on c with $\underline{P} \subseteq P$. Recall that $Q \subseteq I$ is a mongruence on c if and only if $Q \subseteq c^*\mathrm{Pred}(T)(Q)$.

Lemma 8. *The function $\mathcal{P}(I) \to \mathcal{P}(I)$ given by $Q \mapsto c^*\mathrm{Pred}(T)(Q)$ preserves filtered infima $\bigwedge^{\downarrow}$ (with respect to inclusion).*

Proof. By induction on T. The most difficult case is $T(-) = T_1(-) + T_2(-)$. Then, for $Q = \bigwedge^{\downarrow}_{\ell} Q_\ell$ in $\mathcal{P}(I)$, we have $c^*\mathrm{Pred}(T)(Q) = c^* \coprod_\kappa \bigwedge^{\downarrow}_{\ell} \mathrm{Pred}(T_1)(Q_\ell)$ $\cup\, c^* \coprod_{\kappa'} \bigwedge^{\downarrow}_{\ell} \mathrm{Pred}(T_2)(Q_\ell)$, where

$$\begin{aligned} c^* \textstyle\coprod_\kappa \bigwedge^{\downarrow}_{\ell} \mathrm{Pred}(T_1)(Q_\ell) &= \{i \mid \exists x \in T_1(I).\, c(i) = \kappa x \wedge x \in \textstyle\bigwedge^{\downarrow}_{\ell} \mathrm{Pred}(T_1)(Q_\ell)\} \\ &\overset{(*)}{=} \textstyle\bigwedge^{\downarrow}_{\ell} \{i \mid \exists x \in T_1(I).\, c(i) = \kappa x \wedge x \in \mathrm{Pred}(T_1)(Q_\ell)\} \\ &= \textstyle\bigwedge^{\downarrow}_{\ell} c^* \coprod_\kappa \mathrm{Pred}(T_1)(Q_\ell). \end{aligned}$$

The equation $(*)$ holds because we have a filtered meet (so that there is at least one ℓ), and because the coprojections κ, κ' are injective. □

Proposition 9. *For an arbitrary predicate $P \subseteq I$, put*

$$P_0 = P \qquad \text{and} \qquad P_{n+1} = P_n \cap c^*\mathrm{Pred}(T)(P_n).$$

We get a descending chain $(P_n)_{n \in \mathbb{N}}$, so that we can take $\underline{P} = \bigwedge^{\downarrow}_n P_n$.

This $\underline{P} \subseteq I$ is then the greatest mongruence contained in P.

Proof. The predicate $\underline{P}$ is the greatest fixed point $\underline{P} = \bigwedge^{\downarrow}_n \Phi^{(n)}(P)$ of the $\bigwedge^{\downarrow}$-preserving function $\Phi\colon \downarrow P \to \downarrow P$ given by $Q \mapsto Q \cap c^*\mathrm{Pred}(T)(Q)$. From $\underline{P} = \Phi(\underline{P})$ we get $\underline{P} \subseteq P$ and $\underline{P} \subseteq c^*\mathrm{Pred}(T)(\underline{P})$, so that $\underline{P}$ is a mongruence contained in P. And if $Q \subseteq P$ is also a mongruence, then $Q \subseteq c^*\mathrm{Pred}(T)(Q)$, so that $Q = Q \cap c^*\mathrm{Pred}(T)(Q) = \Phi(Q)$. But since $\underline{P}$ is the greatest fixed point of Φ, we get $Q \subseteq \underline{P}$. □

6 Cofree coalgebras without equations

The standard way to obtain the terminal coalgebra of a functor $T\colon \mathbf{Sets} \to \mathbf{Sets}$ is to take the limit of the ω-chain $1 \leftarrow T(1) \leftarrow T^2(1) \leftarrow \cdots$, under suitable cocontinuity assumptions on T. These are always satisfied by the polynomial functors T that come from destructor signatures. The (carriers of the) terminal coalgebras that one gets in this way are sets of (generally infinite) trees, which can be described in terms of observations, see [11]. We mention a particular case (which is essentially as used in [13, 16]).

Proposition 10. *The terminal coalgebra of the functor $X \mapsto A \times (B \Rightarrow X)$ associated with the signature $X \longrightarrow A$, $B \times X \longrightarrow X$ is the set of functions $B^\star \Rightarrow A$, with operations $(B^\star \Rightarrow A) \longrightarrow A$ by $\varphi \mapsto \varphi([\,])$, and $B \times (B^\star \Rightarrow A) \longrightarrow (B^\star \Rightarrow A)$ by $(b, \varphi) \mapsto \lambda\beta.\, \varphi(b \cdot \beta)$.* $\square$

With this result, we know that the terminal coalgebra for the bank account signature $\mathsf{bal}\colon X \to \mathsf{N}$, $\mathsf{credit}\colon \mathsf{N} \times X \to X$ is $\mathbb{N}^\star \to \mathbb{N}$. By suitably rearranging destructors, we can also use the proposition for an extended bank account, with additional destructors $\mathsf{name}\colon X \to \mathsf{M}$, $\mathsf{debit}\colon \mathsf{N} \times X \to X$. The name and balance attributes may be combined into a single attribute $X \to \mathsf{N} \times \mathsf{M}$. Similarly, the credit and debit methods may be combined into a single method $(\mathsf{N} + \mathsf{N}) \times X \to X$. As terminal coalgebra we thus get $(\mathbb{N} + \mathbb{N})^\star \Rightarrow M \times \mathbb{N}$, where $M = [\![\, \mathsf{M} \,]\!]$ is a set of 'names' and $\mathbb{N} = [\![\, \mathsf{N} \,]\!]$.

Theorem 11. *For a polynomial functor $T\colon \mathbf{Sets} \to \mathbf{Sets}$ we get a right adjoint G_0 to the forgetful functor $U\colon \mathrm{CoAlg}(T) \to \mathbf{Sets}$ by*

$$C \mapsto \big[\pi' \circ c \text{ where } c \text{ is the terminal coalgebra of } X \mapsto C \times T(X)\big].$$

Moreover, the forgetful functor U is comonadic: the induced comonad $H_0 = UG_0$ on $\mathbf{Sets}$ has an isomorphism $K\colon \mathrm{CoAlg}(T) \to \mathbf{Sets}^{H_0}$ as comparison functor—where $\mathbf{Sets}^{H_0}$ is the category of coalgebras of the comonad H_0.

Proof. By definition we have $c\colon H_0C \xrightarrow{\cong} C \times T(H_0C)$ and $G_0C = \big(\pi' \circ c\colon H_0C \to T(H_0C)\big)$. For a T-coalgebra $d\colon W \to TW$ there is a bijective correspondence between functions $f\colon W \to C$ and coalgebra maps $g\colon W \to H_0C$. Indeed, given f, one obtains $\overline{f}\colon W \to H_0C$ with $c \circ \overline{f} = id \times T(\overline{f}) \circ \langle f, d\rangle$ by terminality of c. This $\overline{f}$ is then also a coalgebra map $d \to G_0(C)$. And conversely, given a coalgebra map $g\colon W \to H_0C$, one takes $\overline{g} = \pi \circ c \circ g\colon W \to C$.

Comonadicity follows from "co-Beck". But it is not hard to see that the inverse of the comparison functor K maps a (comonad-)coalgebra $\psi\colon C \to H_0C$ to the T-coalgebra

$$C \xrightarrow{\psi} H_0C \xrightarrow{\pi' \circ c} T(H_0C) \xrightarrow{T(\pi \circ c)} T(C). \qquad \square$$

The algebra version of this result is quite familiar: for a polynomial functor T, the *left* adjoint to the forgetful functor $\mathrm{Alg}(T) \to \mathbf{Sets}$ maps a "set of variables" V to the algebra of terms, with variables from V; this algebra is obtained as $v \circ \kappa'$ from the initial algebra v of $X \mapsto V + T(X)$. The resulting adjunction occurs as Proposition 5 in [4, Section 9.4], with this initial algebra computed explicitly as a colimit. In the coalgebra case the set C on which one constructs the cofree coalgebra may be seen as a set of "colours", which are used as extra labels in the trees in the terminal coalgebra of T, see [11].

7 Cofree coalgebras with equations

Assume a destructor signature with $T\colon \mathbf{Sets} \to \mathbf{Sets}$ as associated functor, and with a set of equations, $\mathcal{E} = \{\Gamma_\ell, x\colon X \vdash M_\ell =_{\rho_\ell} M'_\ell \mid \ell \in K\}$. If we also assume a coalgebra $c\colon I \to T(I)$, then we can interprete these terms as maps $[\![M_\ell]\!], [\![M'_\ell]\!]\colon [\![\Gamma_\ell]\!] \times [\![X]\!] \rightrightarrows [\![\rho_\ell]\!]$, where $[\![X]\!] = I$. By abstraction, we get two maps $[\![X]\!] \rightrightarrows ([\![\Gamma_\ell]\!] \Rightarrow [\![\rho_\ell]\!])$, the equalizer of which will be written as $P_\ell \hookrightarrow [\![X]\!]$. Hence $P_\ell = \{x \in [\![X]\!] \mid \forall y \in [\![\Gamma_\ell]\!].\, [\![M_\ell]\!](y, x) = [\![M'_\ell]\!](y, x)\}$. Put $P = \bigcap_\ell P_\ell \hookrightarrow [\![X]\!]$, and let $\underline{P}$ be the greatest mongruence contained in this intersection P. Then $\underline{P} \hookrightarrow I$ carries a sub-coalgebra structure $c_{\mathcal{E}}\colon \underline{P} \to T(\underline{P})$, which satisfies the equations in $\mathcal{E}$. The latter follows from the inclusion $\underline{P} \subseteq P$.

Example 3. Let us reconsider the bank account signature $\mathsf{bal}\colon X \to \mathsf{N}$, $\mathsf{credit}\colon \mathsf{N} \times X \to X$ and its (terminal) coalgebra with carrier $\mathbb{N}^\star \Rightarrow \mathbb{N}$ as in Example 1. Its operations are $\mathsf{bal}(\varphi) = \varphi([\,])$ and $\mathsf{credit}(n, \varphi) = \lambda\alpha.\, \varphi(n \cdot \alpha)$. We consider the singleton set of equations,

$$\mathcal{E} = \{n\colon \mathsf{N}, x\colon X \vdash \mathsf{bal}(\mathsf{credit}(n, x) =_{\mathsf{N}} n + \mathsf{bal}(x)\}.$$

We get $P \subseteq \mathbb{N}^\star \Rightarrow \mathbb{N}$, as above, consisting of those $\varphi \in \mathbb{N}^\star \Rightarrow \mathbb{N}$ with $\forall n \in \mathbb{N}.\, \varphi(n \cdot [\,]) = n + \varphi([\,])$. If we construct the greatest mongruence $\underline{P}$ contained in P as in Proposition 9, then $P_0 = P$ and

$$P_1 = P_0 \cap \{\varphi \mid \forall n_1, n_2.\, \varphi(n_1 \cdot n_2 \cdot [\,]) = n_2 + \varphi(n_1 \cdot [\,])\}.$$

Carrying on like this, we get

$$\underline{P} = \{\varphi \mid \forall \alpha \in \mathbb{N}^\star.\, \forall n \in \mathbb{N}.\, \varphi(\alpha \cdot n \cdot [\,]) = n + \varphi(\alpha \cdot [\,])\}.$$

This means that $\varphi \in \underline{P}$ is completely determined by its value $\varphi([\,]) \in \mathbb{N}$ on the empty list. Hence $\underline{P}$ can be identified with $\mathbb{N}$, and the inclusion $\mathbb{N} \hookrightarrow (\mathbb{N}^\star \Rightarrow \mathbb{N})$ is $n \mapsto \lambda\alpha.\, n + \mathsf{sum}(\alpha)$. The induced coalgebra structure $\mathbb{N} \to \mathbb{N} \times (\mathbb{N} \Rightarrow \mathbb{N})$ on $\mathbb{N}$ is $n \mapsto \langle n, \lambda m.\, n + m\rangle$ as in Example 1 (b). It satisfies the equations.

(Notice, by the way, that the value $n = \varphi([\,]) \in \mathbb{N}$ can be seen as the amount of money put in the bank account at initialization. The resulting account $\lambda\alpha.\, n + \mathsf{sum}(\alpha)$ may be read as $\mathsf{new}(n)$ in OO-terminology.)

The above procedure of obtaining coalgebras satisfying certain equations yields an outcome with a universal property. This is our main result.

Theorem 12. *For a polynomial functor $T\colon \mathbf{Sets} \to \mathbf{Sets}$ we get a right adjoint G_1 to the forgetful functor $U\colon \mathrm{CoAlg}(T, \mathcal{E}) \to \mathbf{Sets}$, where $\mathcal{E}$ is a suitable set of equations. Moreover, this functor U is comonadic.*

Proof. For $C \in \mathbf{Sets}$, we write $c\colon H_0C \xrightarrow{\cong} C \times T(H_0C)$ for the terminal coalgebra as in Theorem 11. Let $H_1C \hookrightarrow H_0C$ be the greatest mongruence induced by the equations in $\mathcal{E}$, as above, and write $G_1C = \big(d\colon H_1C \to T(H_1C)\big)$ for the induced subcoalgebra (which satisfies the equations). For a coalgebra $e\colon W \to TW$ satisfying the equations in $\mathcal{E}$, and a function $f\colon W \to C$, we get a coalgebra map $\overline{f}\colon W \to H_0C$ as in the proof of Theorem 11. Since $\overline{f}$ is a coalgebra map, it commutes with the interpretations of terms (in e and in $\pi' \circ c$). And since the equations hold in e, $\overline{f}$ factors through the intersection of the equalizers induced by the equations in $\mathcal{E}$. But the image of $\overline{f}$ is a mongruence, so it must be contained in the greatest mongruence H_1C induced by this intersection. Hence $\overline{f}$ factors through $H_1C \hookrightarrow H_0C$.

Comonadicity follows from "co-Beck": U creates equalizers for coalgebra maps $u, v\colon I \rightrightarrows J$ which have an absolute equalizer $m\colon I' \rightarrowtail I$ in $\mathbf{Sets}$. By absoluteness, one gets a sub-coalgebra structure $I' \dashrightarrow T(I')$ on I'. It satisfies the equations since it is a subcoalgebra. It is easy to see that $m\colon I' \rightarrowtail I$ is also equalizer of u, v in $\mathrm{CoAlg}(T, \mathcal{E})$. □

We have constructed the cofree coalgebra (satisfying the equations) on an arbitrary set C, with $H_1(C)$ as carrier. In [6] only the cofree coalgebra on the singleton (terminal) set 1 is described, namely as

$$H_1(1) = \bigcup \{\mathrm{Im}(!_c) \subseteq H_0(1) \mid c \in \mathrm{CoAlg}(T, \mathcal{E})\}$$

where $!_c\colon c \to H_0(1)$ is the unique coalgebra map from c to the terminal coalgebra $H_0(1)$.

8 Adding powerset

One possible extension of the above-used notion of polynomial functor is to add the powerset functor $\mathcal{P}$. This may be finite subsets only, so that more terminal coalgebras exist, see [3]. One adds a fifth formation rule in Definition 1 (for $\mathbb{C} = \mathbf{Sets}$),

(v) if $T\colon \mathbf{Sets} \to \mathbf{Sets}$ is a polynomial functor, then so is $X \mapsto \mathcal{P}(T(X))$.

Then we can consider functors like $X \mapsto \mathcal{P}(A \times X)$, coalgebras of which are A-labelled transition systems, see [1, 17]. In our approach we first lift this powerset functor to predicates and relations.

Definition 13. Define the lifted powerset functor $\mathrm{Pred}(\mathcal{P})\colon \mathbf{Pred} \to \mathbf{Pred}$ by

$$(Q \subseteq I) \mapsto (\{a \mid \forall i \in a.\, Q(i)\} \subseteq \mathcal{P}(I)).$$

And $\mathrm{Rel}(\mathcal{P})\colon \mathbf{Rel} \to \mathbf{Rel}$ by

$$\begin{aligned}(R \subseteq I \times I) \mapsto (\{(a,b) \mid \forall i \in a.\, \exists j \in b.\, R(i,j) \\ \wedge \forall j \in b.\, \exists i \in a.\, R(i,j)\} \subseteq \mathcal{P}(I) \times \mathcal{P}(I)).\end{aligned}$$

This lifting is chosen because it satisfies the same properties of lifted functors that we have been using before, so that the same results apply for the extended notion of polynomial functor.

Proposition 14. (i) *The functor* $\mathrm{Pred}(\mathcal{P})$ *is fibred and cofibred:*

$$\begin{aligned}\mathrm{Pred}(\mathcal{P})(u^*(Q)) &= \mathcal{P}(u)^*\mathrm{Pred}(\mathcal{P})(Q) \\ \mathrm{Pred}(\mathcal{P})(\textstyle\coprod_u(Q)) &= \textstyle\coprod_{\mathcal{P}(u)} \mathrm{Pred}(\mathcal{P})(Q).\end{aligned}$$

It also commutes with truth and comprehension:

$$\mathrm{Pred}(\mathcal{P})(1(I)) = 1(\mathcal{P}(I)) \qquad \textit{and} \qquad \{\mathrm{Pred}(\mathcal{P})(Q)\} \cong \mathcal{P}(\{Q\}).$$

(ii) *Similarly,* $\mathrm{Rel}(\mathcal{P})$ *is a fibred and cofibred functor, which commutes with equality.*

(It also commutes with quotients in the sense that $\mathcal{P}(I/R) \cong \mathcal{P}(I)/\mathrm{Rel}(\mathcal{P})(R)$ *for* $R \subseteq I \times I$ *an equivalence relation.)* □

Example 4. Consider a coalgebra $c\colon I \to \mathcal{P}(I \times A)$; it can be understood as an A-labelled transition system where $i \xrightarrow{a} j \Leftrightarrow (a,j) \in c(i)$. A mongruence on c is then a predicate $P \subseteq I$ satisfying for $i \in I$,

$$\text{if } P(i) \text{ then } \forall j \in I.\, \forall a \in A.\, i \xrightarrow{a} j \supset P(j).$$

And a coinduction assumption on c is a relation $R \subseteq I \times I$ such that for $i, i' \in I$,

$$\begin{aligned}\text{if } R(i,i') \text{ then } \ \forall j.\, \forall a.\, i \xrightarrow{a} j \supset \exists k.\, i' \xrightarrow{a} k \wedge R(j,k) \\ \wedge \forall k.\, \forall a.\, i' \xrightarrow{a} k \supset \exists j.\, i \xrightarrow{a} j \wedge R(j,k)\end{aligned}$$

Such a relation is usually called a bisimulation.

9 Further work

We have described some of the basics of coalgebra. Much remains to be investigated, for example whether or not one has: completeness of a suitable logic in coalgebra, a "Birkhoff variety" theorem, or tensors of coalgebras.

References

1. P. Aczel. *Non-well-founded sets.* Number 14 in Lecture Notes. CSLI, Stanford, 1988.
2. P. Aczel and N. Mendler. A final coalgebra theorem. In D.H. Pitt, A. Poigné, and D.E. Rydeheard, editors, *Category Theory and Computer Science*, number 389 in Lect. Notes Comp. Sci., pages 357–365. Springer, Berlin, 1989.
3. M. Barr. Terminal coalgebras in well-founded set theory. *Theor. Comp. Sci.*, 114(2):299–315, 1993. Corrigendum in *Theor. Comp. Sci.* 124:189–192, 1994.
4. M. Barr and Ch. Wells. *Toposes, Triples and Theories.* Springer, Berlin, 1985.
5. W.R. Cook. Object-oriented programming versus abstract data types. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages*, number 489 in Lect. Notes Comp. Sci., pages 151–178. Springer, Berlin, 1990.
6. U. Hensel and H. Reichel. Defining equations in terminal coalgebras. Manuscript, Univ. of Dresden. To appear in E. Astesiano et al. (eds.), *Recent trends in Data Type Specification* Springer LNCS, 1995.
7. C. Hermida. *Fibrations, Logical Predicates and Indeterminates.* PhD thesis, Univ. Edinburgh, 1993. Techn. rep. LFCS-93-277. Also available as Aarhus Univ. DAIMI Techn. rep. PB-462.
8. C. Hermida and B. Jacobs. Induction and coinduction via subset types and quotient types. Manuscript available from `ftp.cwi.nl` in `/pub/bjacobs`, December 1994.
9. C. Hermida and B. Jacobs. An algebraic view of structural induction. To appear in proceedings of *Computer Science Logic 1994.* Springer LNCS, 1995.
10. M. Hofmann and B. Pierce. An abstract view of objects and subtyping. *Journ. Funct. Progr.*, 1995.
11. B. Jacobs. Bisimulation and apartness in coalgebraic specification. Manuscript available from `ftp.cwi.nl` in `/pub/bjacobs`, January 1995.
12. B. Jacobs. Parameters and parametrization in specification using distributive categories. *Fund. Informaticae*, 1995.
13. S. Kamin. Final data types and their specification. *ACM Trans. on Progr. Lang. and Systems*, 5(1):97–123, 1983.
14. S. Mac Lane. *Categories for the Working Mathematician.* Springer, Berlin, 1971.
15. E.G. Manes. *Algebraic Theories.* Springer, Berlin, 1974.
16. H. Reichel. An approach to object semantics based on terminal co-algebras. *Math. Struct. Comp. Sci.*, 1995.
17. J. Rutten and D. Turi. On the foundations of final semantics: non-standard sets, metric spaces and partial orders. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Semantics: Foundations and Applications*, number 666 in Lect. Notes Comp. Sci., pages 477–530. Springer, Berlin, 1993.

This article was processed using the LaTeX macro package with LLNCS style